AUTOMATED CARTOGRAPHIC NAME PLACEMENT USING RULE-BASED SYSTEMS

By

Anthony Charles Cook B.Sc. M.Sc.

Thesis submitted to the CNAA in partial fulfilment of the requirements for the degree of Doctor of Philosophy

The collaborating establishment was
The Ordnance Survey, Southampton

Department of Mathematics and Computer Science
The Polytechnic of Wales
Pontypridd, Mid Glamorgan
CF37 1DL

SEPTEMBER 1988

CONTENTS

6 "LABPOS" - A PROTOTYPE NAME PLACEMENT SYSTEM

7 "NAMEX" - A RULE-BASED NAME PLACEMENT SYSTEM

8 RULE-BASED NAME PLACEMENT USING THE NAMEX SYSTEM

9 CONCLUSION TO THE THESIS

APPENDICES

REFERENCES

ACKNOWLEDGEMENT

I would like to thank my supervisors Dr. C.B. Jones (Polytechnic of Wales), Dr. D.H. Smith (Polytechnic of Wales) and Mr. D.W. Proctor (Ordnance Survey) for their invaluable discussions during the research and for their guidance during the thesis write-up.

My thanks also go to members of staff and colleagues at the Department of Mathematics and Computer Science for their friendship and useful discussions.

The practical aspects of this research have been aided by the computer centre staff who have always been willing to answer my technical questions on computing.

I would also like to thank my parents for putting up with me at the weekends and also for their support and encouragement.

Finally my thanks go to the research staff of the collaborating body, the Ordnance Survey, for their keen interest in the research and to the Science and Engineering Research Council for sponsoring me for the first couple of years of the PhD.

Certificate of Research

This is to certify that, except where specific reference is made, the work described in this thesis is the result of the investigation carried out by the candidate.

Candidate A.C.Cook.

Director of studies C.B.Jones

Declaration

This is to certify that neither this thesis nor any part of it has been presented or is being concurrently submitted in candidature for any other degree than the degree of Doctor of Philosophy of the CNAA.

Candidate A.C.Cook.

"AUTOMATED CARTOGRAPHIC NAME PLACEMENT USING RULE-BASED SYSTEMS"

BY ANTHONY CHARLES COOK

ABSTRACT

This thesis describes automated cartographic name placement using rule-based systems. In particular it describes the problem involved with designing a system which is flexible enough to place names on a variety of maps. This is demonstrated using logic programming techniques written in PROLOG.

Most previous name placement systems are either map specific or have demonstrated only a few aspects of name placement. However two of these systems, which use the rule-based approach for solving the name placement problem, do show greater flexibility. Nevertheless all known results from these seem unsophisticated when compared to many manually produced maps. This thesis describes further research into the use of rule-based systems. The systems described have the capability to handle a wider range of maps of greater complexity. Also described is a procedural program which implements an iterative strategy for name placement on the Ordnance Survey Route Planner map.

The research attempts to classify label positions and configurations used on a wide range of maps and discusses ways of implementing these in an automated name placement system. A range of name placement rules are also studied in order to decide what type of data a flexible automated name placement system must be able to access. A combined vector and raster data structure approach is adopted. This supplies the necessary "visual" information needed to apply most of the name placement rules. Name placement and database primitives are used to construct the high level rules which make up the rule-based systems.

This work has been undertaken in collaboration with the Ordnance Survey. The procedural name placement program, capable of placing names on the 1:625000 Route Planner map, has been implemented at their headquarters.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Although many aspects of map production have proved capable of automation since the early 1970's, automated name placement has only been tackled with vigour comparatively recently (Freeman and Ahn, 1984). The positioning of names on a map is both a complex and lengthy task which has been estimated to take up to 50% of the manual production time for completely new maps (Imhof, 1975). On the Ordnance Survey Route Planner map for instance, a cartographer typically places names at a rate of 20 per hour on a fully revised edition. For other maps, this rate may be as high as 30 names per hour (Anon, O.S.). Performing this process automatically could significantly reduce the time taken.

After discovering that many of the existing name placement systems (Chapter 4) were designed for specific maps, the author decided upon an aim which was to produce an automated rule-based name placement system which could be used to place names on a wide variety of maps providing that the rules were explicitly defined. Further fundamental problems in existing automated name placement

systems which remain to be solved are listed in section 4.6. Another aim of this present research was to develop an automated rule-based name placement system which would be capable of working on the VAX 11/750 computer of the collaborating body (Ordnance Survey) using facilities that they either had available or were willing to acquire.

In any form of map making, the placing of names has three major constraints. Firstly names must be placed so that they do not overlap each other or important underlying features. Secondly names must be placed to avoid ambiguity over which cartographic feature they represent. Thirdly names which have been placed must look aesthetically pleasing on the map. The relative importance of cartographic names and features and other considerations, dependent on map type, govern the application of these three constraints. General principles of map annotation are discussed further in chapter 2.

On first impression, attempting to perform the name placement task by means of a computer may seem an obvious development and quite straightforward. However, in practice the problem is not as simple because of two distinct advantages the human cartographer has over the computer. Firstly, he has a superb real time parallel vision system, and secondly a huge geographical knowledge base from which various name placement rules and information can be extracted. As yet, no computer vision

system comparable with that of the human visual system has been developed, nor can the same amount of geographical expertise that a human expert possesses in his brain be encoded into a computer (this is because there are a very large number of rules used and the difficulty involved with extracting all of these from experts). With these two major computational deficiencies, how can the name placement problem be solved using a computer?

In order to position names automatically on a map, the advantages that computers have over humans must be exploited. These are that computers are very fast at numerical work, and can quickly access specific map features from a structured cartographic database. They are also able to emulate, to some extent, the human capacity for deduction from known facts and rules. Based upon these important advantages, an automatic name placement system has been constructed which achieves results of consistent quality, and faster than can be achieved manually. The speed and efficiency of this automated process depends strongly upon the efficiency of the program, the computational power and computer architecture available.

The research into automated name placement undertaken by the author has been carried out in collaboration with the Ordnance Survey (see section 1.2). The Ordnance Survey supplied cartographic data in the form of the Route Planner map database which, due to its higher than average feature and name density, formed an appropriate test map for an automated name placement system.

1.2 THE ORDNANCE SURVEY AND DIGITAL MAPPING

The Ordnance Survey is a British cartographic organisation which has been in existence since 1791 and was originally set up to coordinate and control official surveying and topographic mapping throughout Britain. The Ordnance Survey is now an independent body within the Department of the Environment and has the task of recording every factory and building built or demolished in the entire country (Paulframan, 1984).

Computer cartography began as long ago as the 1950's (Monmonier, 1982) with the production of crude maps generated on electric typewriters connected to computers. However in the 1960's when slightly more sophisticated hardware became available, such as line printers, and later on cathode ray tube displays and digitisers, computer assisted cartography became a practical tool. Software such as SYMAP and CMAP were written to make use of the hardware. The SYMAP system, written in 1963 by Fisher (Burrough, 1986), utilised the overprint facility on the lineprinter to produce maps containing either statistical values or corresponding grey levels. This was so successful amongst geographers that it soon became a standard and is still used (Nagy and Wagle 1979). The 1970's saw various cartographic organisations introduce computer techniques to assist in map making with the aim of helping map revision work and reducing the cost of

production.

The 1980's is a decade in which further automation is taking place exemplified by automatic digitisation using line following devices and raster scanners, line generalising techniques and the use of expert systems in several aspects of map production. Also, automated cartography is becoming the rule rather than the exception in several cartographic organisations.

The Ordnance Survey began experimenting with computer assisted cartography in 1968 with emphasis placed on the large scale 1:1250 and 1:2500 maps (Mayes, 1986). Digital map production began in the mid 1970's with the aim of digitising all 220,000 large scale British maps. A cartographer typically spends two days digitising a complete map, though this obviously depends upon the amount of detail present (Paulframan, 1984). Complete digitisation of 1:1250 large scale maps of major urban areas is expected to be accomplished by 1995 and all other warranted areas within the following decade (Bowell, 1988). This is one of the major bottlenecks mentioned in the previous section and is a sought after goal in cartographic research.

Maps digitised by the Ordnance Survey include the 1:625000 small scale Route Planner, 1:1250 and 1:2500 large scale maps and one or two sheets from 1:10000 and

1:50000 scale maps (Bowell, 1988). The Route Planner map database and software were developed over the period 1980 to 1985 to gain experience before tackling databases for large scale maps where the amount of data and subsequent risk are greater (Haywood, 1984).

The Ordnance Survey produced their first commercial small scale digital map, the Route Planner, in 1986. Names were initially placed automatically in the vicinity of the map feature they referred to and manual name placement was performed with the aid of an interactive graphics screen and tracker ball. Each name position, orientation, and size could be edited according to the cartographer's choice (Hadley, 1986).

A good commercial reason for developing a name placement system to place names on the Route Planner road map is that road map marketing is a very competitive business in Britain with many cartographic and motoring organisations producing their own maps in order to persuade the lucrative motoring market to purchase up-to-date maps of the road network. With an estimated 20 million motorists in Britain (Robbins, 1985) and the number of new motorists increasing by approximately 1 million per year (HMSO, 1979), any system which enables cheaper and faster means of updating road maps would be of great financial benefit to a cartographic organisation. Normally, the entire process of road map publication may

take anywhere between 18 months and three years and is very costly. The compiling, drawing and photography involved in the production of a 4 colour small scale map of reasonable density at A4 size can cost approximately £2000. Therefore map publishers prefer to use the same basic maps over several years of publication with only a few modifications where necessary, for instance an occasional change in presentation or perhaps an alteration of scale. In this relatively unchanged format, a map may be produced for several editions for up to 25 years (Robbins, 1985).

In view of the enormous time and effort involved in the process of digitising maps, the Serpell report by the House of Lords select committee on Remote Sensing and Digital Mapping in early 1984, recommended the Ordnance Survey to search for a cheaper and faster means of digitising maps. Collaborative work with Universities and Polytechnics is one means of achieving this goal (Rhind, 1985). The research into the name placement problem, although not connected with digitisation, forms part of an effort to speed up and reduce the cost of map production.

1.3 LOGIC PROGRAMMING

In order to attempt to write a computer program that will be able to position names on a map automatically, the complicated process must be broken down into a well defined set of rules. A logic program consisting of such a set of rules is said to be "rule-based". The rules are of the form "Conclusion A if Conditions B, C, D, ... are true". Conclusion A is the goal which is the logical consequence of a finite set of rules which depend upon available facts or upon the results of their predecessors.

This programming approach was selected for the name placement system because it had several distinct advantages over conventional programming techniques. One benefit is that logical rules can, when written in say the language of PROLOG (PROgramming in LOGic), have the ability to backtrack on decisions. Backtracking is used in manual name placement, where for instance if a name has been placed and this blocks the placement of other names later on, then the original decision may be reversed and the blocking label placed elsewhere (Anon, O.S.). PROLOG is a descriptive language and programs written in it can be made very modular and hierarchical in appearance which makes them fairly readable. They can also be very short, for instance a recursive algorithm to solve the Towers of Hanoi problem written in PROLOG needs only four logical clauses (Sterling and Shapiro, 1986), whereas using a

conventional language such as PASCAL requires seventeen lines of code (Graham, 1984). Finally, PROLOG like its main predecessor and competitor LISP, is a widely used artificial intelligence language. However, unlike LISP which is primarily a list processing and symbol manipulation language, PROLOG is essentially an all purpose query language with a built in database structure consisting of rules and facts which make it highly suitable for implementing rule-based tasks.

PROLOG was developed in Marseilles by Colmerauer in the early 1970's. It was based upon Robinson's resolution principle (Robinson, 1965) and Kowalski's problem solving interpretation of statements of declarative logic as procedural instructions to the computer (Kowalski, 1979). It has since become a very popular language amongst artificial intelligence workers in Europe and is gradually being introduced into the U.S.A. where the older language, LISP, is still widely used. The version of PROLOG used for this research was one of three languages contained in the POPLOG package developed at Sussex University. The dialect is the popular Edinburgh PROLOG with a few enhancements, such as the ability to access external subroutines.

Another term often used in the field of artificial intelligence is "expert system". This is a computer program with the built-in knowledge, in a specific subject, extracted from an expert (Waterman, 1985). It is

composed of a knowledge base and an inference engine. A "knowledge base" is a set of facts and executable rules extracted from the expert. The "inference engine" coordinates the way the rules are used, for instance their order of execution and how to apply them to deduce new facts and rules.

A good introduction to logic programming and PROLOG is "The Art of PROLOG" (Sterling and Shapiro, 1986). Appendix 2 additionally gives a brief introduction to the basic concepts of PROLOG.

1.4 PLAN OF RESEARCH AND OVERVIEW OF SUBSEQUENT CHAPTERS

Before designing a name placement system capable of placing names on a wide variety of maps, it was necessary to establish the range of maps available and the types of labels involved. Name placement rules as used by cartographers also had to be extracted. This was to be done by spending some time interviewing cartographers at the Ordnance Survey in Southampton, by visually studying a variety of different types of map and by consulting published rules. The results of this investigation are given in chapter 2.

A suitable data structure capable of supplying the information required by name placement rules had to be devised. Chapter 3 describes a variety of means of storing cartographic vector and raster data. From these, the most suitable vector and raster formats for use with a name placement database were selected. Emphasis was placed on keeping data storage requirements low but allowing for fast access to topological information used in typical name placement rules.

A review of known placement methods needed to be undertaken to see which techniques could prove suitable in the context of the author's research. The disadvantages and merits of label selection, placement algorithms, overlap detection and different strategies for name

placement are discussed (Chapter 4).

Practical methods of parameterizing names and placements for an idealised name placement database are discussed in chapter 5. This takes into account the need to allow for most types of name, but must also restrict the number of possible positions in order to reduce the complexity of name placement decisions.

To evaluate the practicality of some new and existing name placement rules and algorithms, a prototype procedural name placement system ("LABPOS") was constructed using the FORTRAN programming language (Chapter 6). From the experience gained, a logic program based name placement system capable of placing names on a wide variety of maps was devised. This used some of the methods discussed in chapters 3, 4, 5 and 6. The system, called "NAMEX", utilises a combination of a rule-based system (high level name placement rules), written in the logic programming language PROLOG, and fast numerical FORTRAN subroutines (low level rules and primitives to access cartographic data) to yield good name placement solutions (Chapter 7).

Once the NAMEX rule-based name placement system had been developed, it required demonstration on some widely different types of map. Chapter 8 describes these demonstrations as well the implementation in PROLOG of a

new backtracking name placement strategy.

Finally a conclusion to the work described in this thesis is given (chapter 9) with suggestions for improvements and for new fields of research for future name placement systems. Program documentation is presented in a separate volume to this thesis.

1.5 CHAPTER SUMMARY

This introductory chapter describes the complexity of name placement and indicates how useful an automated name placement system could be to map production in terms of efficiency and flexibility. It gives a brief account of the development of computer cartography and the Ordnance Survey's exploitation of this field. It also introduces logic programming, explains why this is useful and describes some of the terminology involved. Finally a plan of research is given.

This thesis should give the reader a good impression of the problems associated with automatically annotating maps and will review previous attempts to tackle the problem. The thesis is intended to provide useful information for potentially advanced users and developers of this and similar types of name placement systems. It may also prove useful reading material for people constructing cartographic expert systems using PROLOG. This is because the PROLOG primitives used, in the automated name placement system, illustrate the amalgamation of a rule-based system with a cartographic database.

CHAPTER 2

MANUAL NAME PLACEMENT

2.1 MAP TYPE AND NAME USAGE

2.1.1 INTRODUCTION

This chapter will show the versatility required by a name placement system if it is going to be able to place names on a wide variety of maps. This section (2.1) of the chapter illustrates the variety of maps which such a system would have to cope with. The next section (2.2) describes a small selection of name placement rules used and the following section (2.3) shows how to analyse the preference of label positions and configurations.

A map can be defined as the scaled representation of features on the geometrically defined surface of an object. Although we shall principally be concerned with the Earth as the object to be mapped, maps can also be produced for the Moon, planets and the celestial sphere. The information represented on a map can be linear which usually relates to boundaries or communications, areal which for example might indicate land use, point like which allows information to be associated with a spot location, and typographic which can give the name of or

additional information about almost any feature. It is the usage of name placement that will be discussed for a variety of map types.

Maps generally fall into two categories: topographic and thematic. Topographic maps are designed to represent as many cartographic features as possible at a particular scale and are therefore multi-purpose. Thematic maps usually represent a single theme such as geology, soil use or vegetation and sometimes use topographic maps as the base map so that the map user can relate to a reference framework (Lawrence).

Map sheets typically contain between several hundred and several thousand names (Imhof). The Ordnance Survey maps covering the whole of Great Britain at a scale of 1:50000 contain an estimated quarter of a million place locations alone (Rhind, 1985). The usage of different types of text style, size, distribution and name placement protocols can greatly affect the appearance of the map. Too many labels can cause ambiguity and lead to confusion in map interpretation, therefore cartographers take great care and attention during name selection, compositing and placement stages.

To label a map initially requires the provision of a list of names associated with selected features. This list is usually compiled by the map surveyor consulting

landowners or administrative authorities, or by questioning local inhabitants in the region concerned. Allowances can be made for local pronunciations and spellings to be included on the map, at the surveyors discretion. In circumstances where names have been changed or the map is of a foreign country, alternative name forms can be given. Consideration is also sometimes made for the inclusion of names of social, political or historical importance. If a small scale map is derived from existing large scale maps, it is almost certainly not practical to include all the names on the new map, therefore names must be selected according to their relative importance. During the revision of maps, a similar process must occur if old names have lessened in importance or there are new features to be named (Keates).

Generally, when labelling maps, the better quality maps are produced with the aid of a draft copy. The draft copy can either be set out on a blueprint of the original map or on a transparent overlay sheet on top of the current map. The draft copy, together with an accompanying names register, allows name selection, font style, label size, label coverage and importance to be decided upon at an editorial stage and also helps to identify mistakes before the final copy is produced (Imhof).

The cartographer has a variety of means of representing a name, but there is an overriding

requirement that all names must be legible. Therefore letters which are narrow and tall with short ascenders and descenders are preferred since they occupy smaller areas and therefore present few problems during placement (Keates). With regard to the selection of label font style, on topographic maps, town and place names are printed with upright letters, italics are chosen for natural features such as seas and rivers, and gothic type can be used for anything of antiquity (Lawrence). The cartographer can select the label font type, size and line thickness weighting in order to emphasize importance. The character font size is defined by the standard point system, a point equalling 0.351mm, and the range of font sizes chosen normally cover 6 to 36 points at intervals of two or more points. Finally, label colour may be selected according to the feature that is represented and legibility (Keates).

Many techniques have been used for applying names to an underlying map. Originally, this was performed by writing the names by hand, but due to its slowness and inconsistency, was superseded by the introduction of stencils, pre-printed letters and type-setting or photo-lettering machines (Keates). During the printing process, a photographic unsharp masking technique is sometimes applied. This enables names to be distinguishable from the background map by masking out detail in a narrow border around where the names would lie

on the underlying map component sheet.

The style of a map is defined by its specifications, layout, lettering, colouring and production method. The next section of the chapter will consider a variety of map types and styles which, although they do not form a complete set of examples, will illustrate the wide range of name selection and name usage. The maps to be discussed are mainly thematic although topographic maps will also be considered.

2.1.2 PLANNING MAPS

These typically range from a scale of 1:500 to 1:50000 and include large scale cadastral and survey maps as well as smaller scale town and city street plans. At scales larger than 1:10000,generalisation of map features does not normally apply, and so usually everything is represented proportionally by its width and size. For smaller scale maps, such as street maps, generalisation begins to be required in order to maintain map readability with respect to road and building outlines.

The largest scale official British topographic maps are 1:1250 and are of town and urban areas. 1:2500 scale maps deal with all areas of Great Britain other than mountains and moorland and 1:10000 scale maps cover

remote areas and include relief information in the form of contours (Lawrence).

On large scale planning maps, land parcel reference numbers and associated land areas are written in the areas concerned. Road number labels are not extensively used, but those present are written using heavy weighting and tend to be placed near to junctions and towards the edge of the map. A more commonly used road label is its name which is normally written in capital letters within the road casing. County and district boundaries are labelled at regular intervals along the appropriate side of the boundaries.

Most of the features represented on large scale planning maps are areal in nature such as buildings. There is often sufficient room for names to be written completely inside these areas if necessary and therefore there are very few problems with labels overlapping other labels which can occur at smaller scales. The larger the map scale, the more information can be included inside areas, for instance, a church illustrated at a scale of 1:10000 scale may contain just the church name whereas, at a scale of 1:2500, both the name and the religion of the church can be given (Fig 2.1).

Town and city street maps such as the popular AZ London (Geographers' A-Z Map Co. Ltd.) are commercially

published planning maps. As the name of these maps infer, the most commonly labelled feature are streets whose names are placed as on large scale planning maps, but are clearly written in prominent capital letters to aid identification. Because the primary users of street plans are people attempting to find a particular location, a gazetteer of street names is usually provided on the reverse of the map or in an index. Additionally, street address numbers are sometimes given to help identify which end of a road an address is at (Fig 2.2).

Unlike most other maps, overlap and intersection between labels is acceptable on street maps where, due to the high information content, no alternative placement is available. This typically occurs when dealing with small sections of roads not capable of taking the whole label or in the vicinity of road junctions or with place and regional names.

Most planning maps are produced in black and white because they are the only colours needed to represent the main cartographic features which basically consist of bounded areas. However on smaller scale planning maps, such as the congested 1:50000 scale Gousha Los Angeles and Hollywood street map (Fig 2.3), the use of colour is essential to help identify map features.

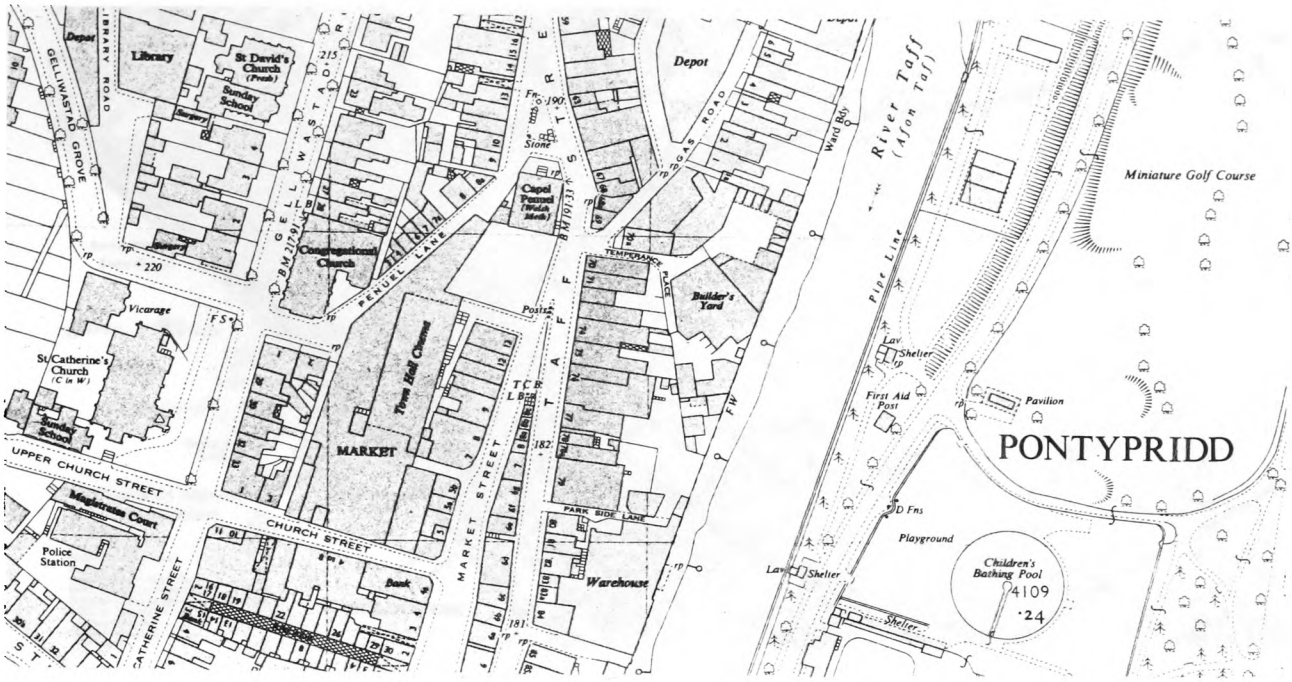


Fig 2.1 An Ordnance Survey Large Scale Plan (Scale 1:2500).

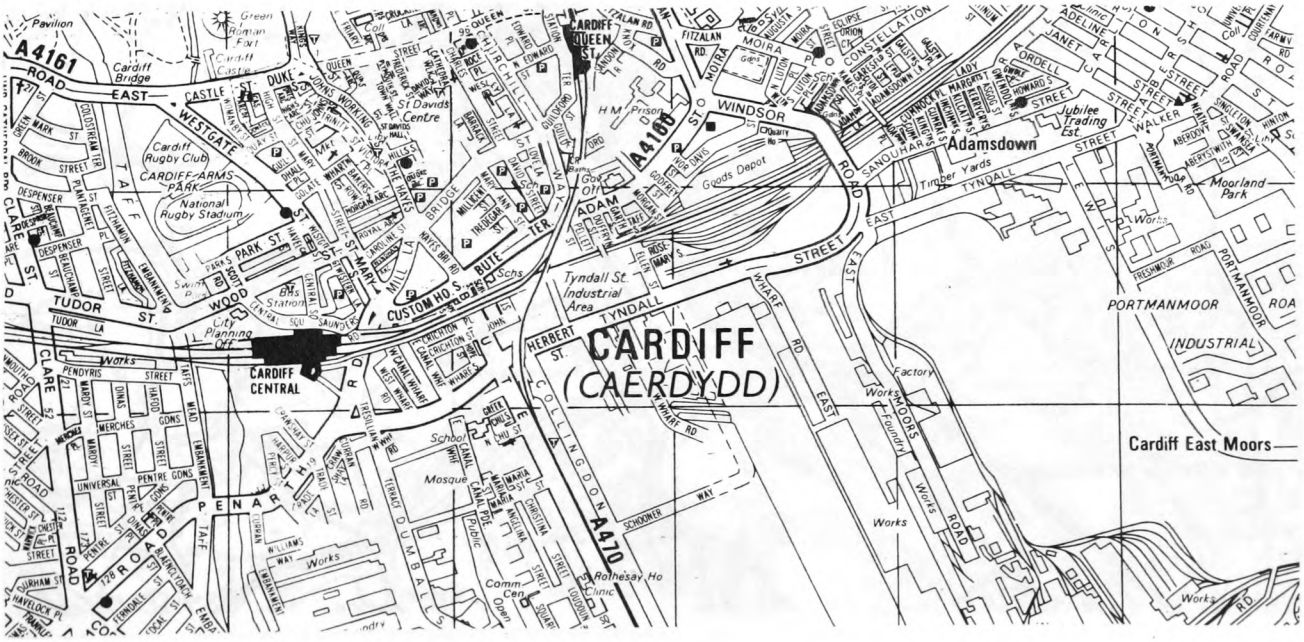


Fig 2.2 Geographers' AZ Street Plan of Cardiff (Scale 1:18103).

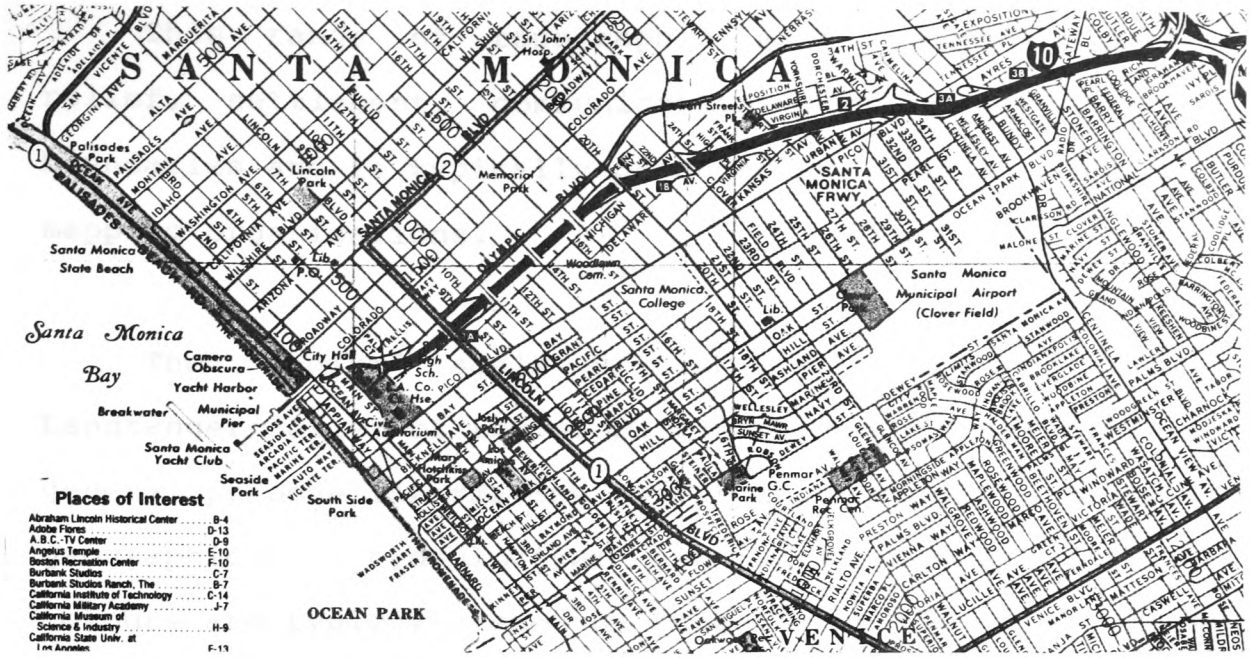


Fig 2.3 Gousha Street Map of Los Angeles and Hollywood (Scale 1:50000).

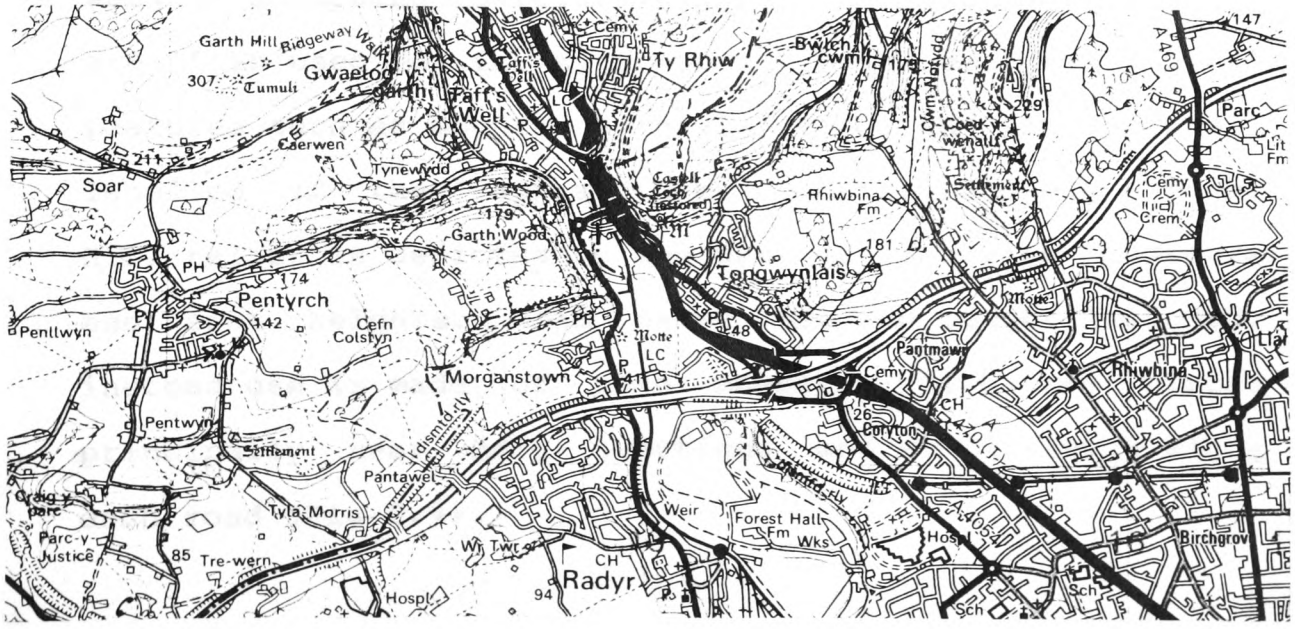


Fig 2.4 A typical "medium scale" map: the Ordnance Survey Landranger Series Map Sheet 171, 1983 (Scale 1:50000).

2.1.3 "MEDIUM" SCALE TOPOGRAPHIC MAPS

This range of topographic map generally depict relief, settlement, communication, vegetation, names and boundaries and is typically produced by official national mapping organisations.

The "medium" scale British Ordnance Survey 1:50000 Landranger series of topographic maps contain a variety of general purpose information concerning motoring, touring, geographical, hydrological and urban contexts. Eight colours are typically used on these maps to distinguish the different variety of features, however only four colours are used for labelling purposes.

Place names vary considerably in letter size and weight according to relative importance, and abbreviations are often used for farms, schools, hospitals etc. Areal features such as mountains, hills, woods and parks are all labelled if they appear to be important and big enough. Hill relief is depicted by the use of labelled contours and spot heights. Road names are no longer included, instead use is made of road numbers which are placed principally near main junctions and at intervals along each road (Fig 2.4).

Finally, light blue British national grid reference numbers are placed at regular intervals on selected

horizontal and vertical grid lines on Landranger maps. These grid numbers can sometimes be offset so as not to overlap with labels and features of the same colour.

2.1.4 ROUTE PLANNING MAPS

This range of maps is produced commercially and aimed at road users and tourists. They typically cover scales of 1:50000 and smaller. Many of the name placement rules used in larger scale maps are not very relevant on these smaller scale maps due to the problems associated with finding free space.

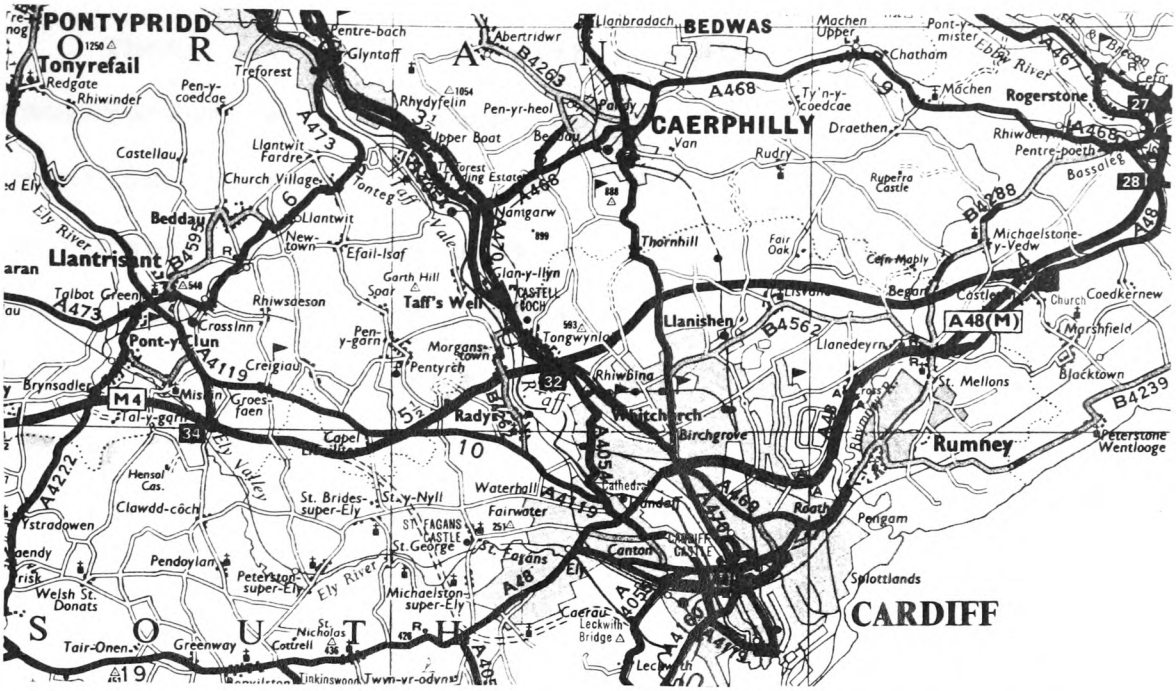
Roads are clearly labelled, as are place names, since these are vital navigational features to road users. Settlement names usually have a standard size according to feature type, except in the case of cities where the labels usually vary in size according to importance. Other features such as rivers and railways are not always represented due to their lower navigational relevance, but important lakes and relief features are retained for naming. Car ferry routes are often included in the form of dashed lines connected to ports and are usually labelled with destination and travel times.

Some route planning maps are specifically designed for touring and therefore contain additional named

features such as camp sites, beaches, castles, golf courses and places of interest. Motoring organisations usually produce this class of maps, which typically cover scales of between 1:50000 and 1:250000 (Fig 2.5).

2.1.5 ATLAS MAPS

All atlases have their own individual styles, but most are intended to portray geography. On world atlases, although the major roads are sometimes shown these are not usually labelled, however main rivers are labelled since these are considered to be important geographical features. Major shipping routes are labelled, presumably because of the large availability of free space at sea. Geographical regions are a very important aspect of atlases and so are labelled with large spread out letters (Fig 2.6).


PONTYPRIDD
Tonyrefail
Llantrisant
Beddau
Pont-y-clun
CAERPHILLY
BEDWAS
Rogerstone
Rumney
CARDIFF
Llanishen
Thornhill
Whitchurch
Radyr
Taff's Well
Castellau
Church Village
Treforest
Rhiwbina
Llanedeyrn
St. Mellons
Splottlands
Penylan
Roath
Canton
Fairwater
Waterhall
Pentyrch
Morgans-town
Michaelston-super-Ely
St. Brides-super-Ely
Peterston-super-Ely
Pendoylan
Welsh St. Donats
Clawdd-coch
Ystradowen
Hensol Cas.
Tair-Onen
Greenway
Cottrell
Rhydyfelin
Pen-yr-heol
Upper Boat
Nantgarw
Efail-Isaf
Talbot Green
Rhiwsaeson
Cross Inn
Creigiau
Groes-faen
Capel Llanilltern
Peterstone Wentlooge
Marshfield
Coedkernew
Blacktown
Michaelstone-y-Vedw
Rudry
Draethen
Machen
Chatham
Pont-y-mister
Ebbw River
Rhiwderin
Pentre-poeth
Bassaleg
Cefn Mably
Begam
Castle
Church
Pentre-bach
Glyntaff
Abertridwr
Llanbradach
Machen Upper
Ty'n-y-coedcae
Pen-y-coedcae
Redgate
Rhiwinder
Treforest
Llantwit Fardre
Tonteg
Llantwit
New-town
Efail-Isaf
Garth Hill
Spar
Taff's Well
Castell Coch
Tongwynlais
Pen-y-garn
Pentyrch
Morgans-town
Rhiwbina
Whitchurch
Birchgrove
Radyr
Fairwater
Waterhall
St. Fagans Castle
St. George
Canton
Cathedral Road
Roath
Penylan
Splottlands
Caerau
Leckwith
Leckwith Bridge
Ely River
Ely Valley
Taff Valley
A470
A48
A48(M)
A4119
A4232
A4222
A473
A468
A469
A472
A4054
A4160
A4161
A4232
A4119
A4232
A4232
A4232

2.1.6 ADMINISTRATIVE AREA MAPS

These are concerned with the display of boundaries of administrative areas such as countries, island groups, counties, districts and political constituencies. A topographic map is usually displayed in the background of such maps. Many administrative areas are hierarchical in nature in that a country can consist of counties which in turn consist of districts and so on. Area hierarchy is portrayed in the associated labels by making use of different weighting and lettering size (Fig 2.7).

2.1.7 MARITIME AND AIR NAVIGATION MAPS

Maritime maps or hydrographic charts were found to have the most varied range of labels encountered by the author during his research. Such maps are designed specifically for use at sea and so land formations are of secondary importance, except for principal ports, river mouths and bays etc. Prominent land features visible from sea such as masts and hills have spot heights labelled in metres and in some cases the spot height label is amalgamated with an abbreviated name describing the feature (Fig 2.8).

Potential shipping hazards such as wrecks, islands, rocks, and low tide sea depth soundings are clearly

labelled. Light houses have the time interval between flashes included and further specifications to allow an unambiguous identification from offshore. Other coastal features labelled include fog horns, buoys, castles, churches, and geographical features all of which usually include abbreviations or a combination of abbreviation and the name. Few linear feature labels are present on nautical maps, for instance there are no roads, however short lengths of rivers near ports are labelled since these are essential navigational features.

Other maritime labels include shipping zones, geographical sea areas, trenches, reefs and bays. There are also large magnetic compass roses present with the bearings and additional information on magnetic variation and direction. Tables are a special form of multi-lined label and these are usually placed in free areas, at the corners or edges of the map. Use is also made of appropriate inland regions which can contain tables of coastal radio station and marine radio beacon frequencies.

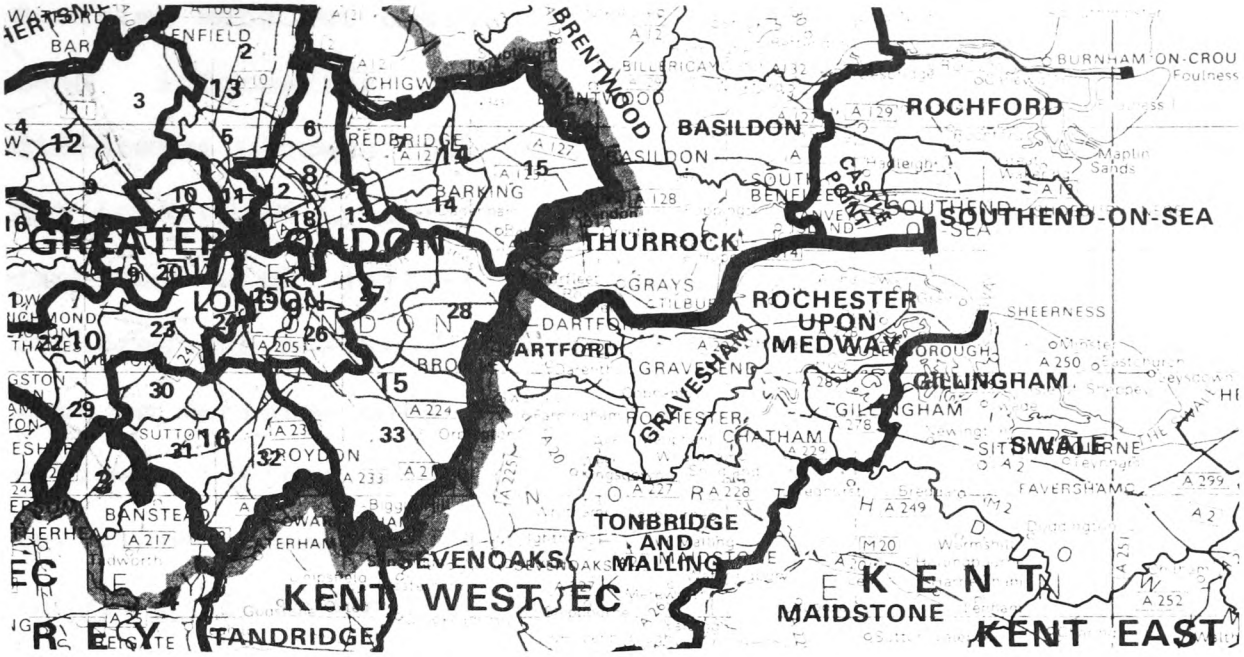


Fig 2.7 Ordnance Survey Administrative Area Map, 1983 (Scale 1:625000).

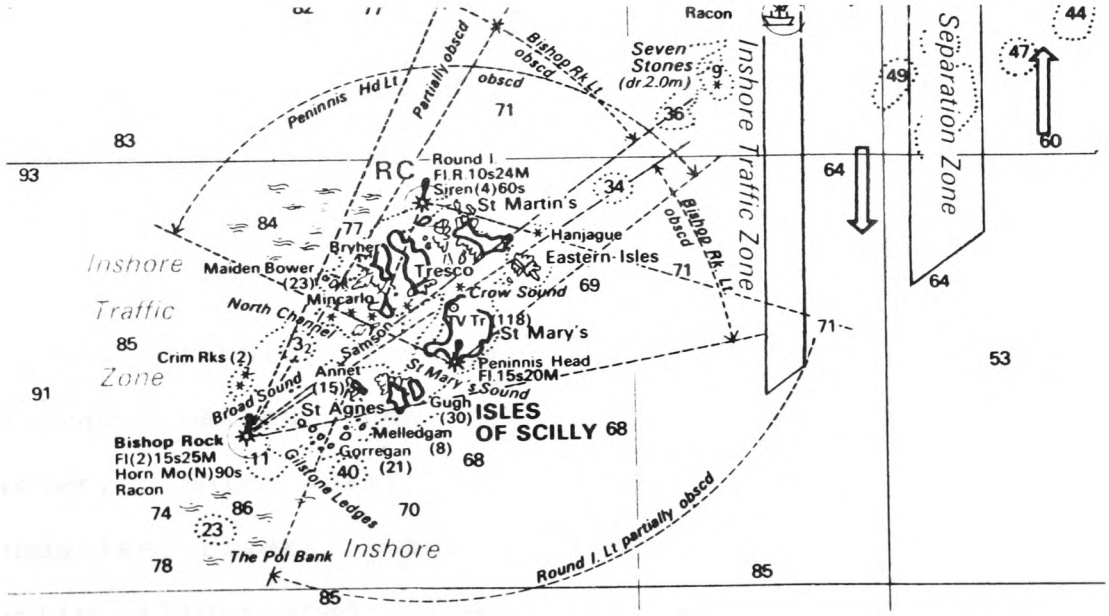


Fig 2.8 Imray C10 Yachting Chart of the Western English Channel, April 1983, (Approximately 1:400000 scale).

Aeronautical maps are similar in certain respects to maritime maps but obviously must contain considerably more land information for flights over land. One major difference between this type of map and others is that the map is divided up into grid cells of air space which are labelled with letters and numbers for quick location of features and position. Additional label information to that on maritime maps concerns contours, air corridors and minimum flying heights. Important information such as restricted areas and flight lanes must be prominently labelled in order that ambiguity does not occur which may impair flight safety (Fig 2.9).

2.1.8 PHOTOMAPS

This class of map tends to be produced for regions which have been poorly mapped, but for which aerial imagery exists and can be used for producing mosaic or airbrush relief maps.

Cartographic features are often superimposed on photomaps making them similar to topographic maps. However, only basic cartographic features such as boundaries, rivers, roads, contours and spot heights are usually illustrated. Since the map is effectively a picture, most of the lettering is kept consistent and small so as to avoid detracting from the pictorial

information.

A rather unusual kind of photomap is that produced for the Moon and planets where the most numerous features are craters. In the case of heavily cratered worlds such as the Moon, there is a large degree of freedom in labelling since relatively few features have names and because label size tends to vary with feature dimensions. Another unusual aspect is that minor craters in the vicinity of main craters have the name of the main crater suffixed by a unique letter identifier. When written on the map, minor craters can either be written with the full name or just the letter. Although the degree of freedom of name placement on planetary terrains is high, there are two major constraints, firstly that letters must not cover any underlying features, and secondly ambiguity leading to misidentification of features must be avoided (Fig 2.10).

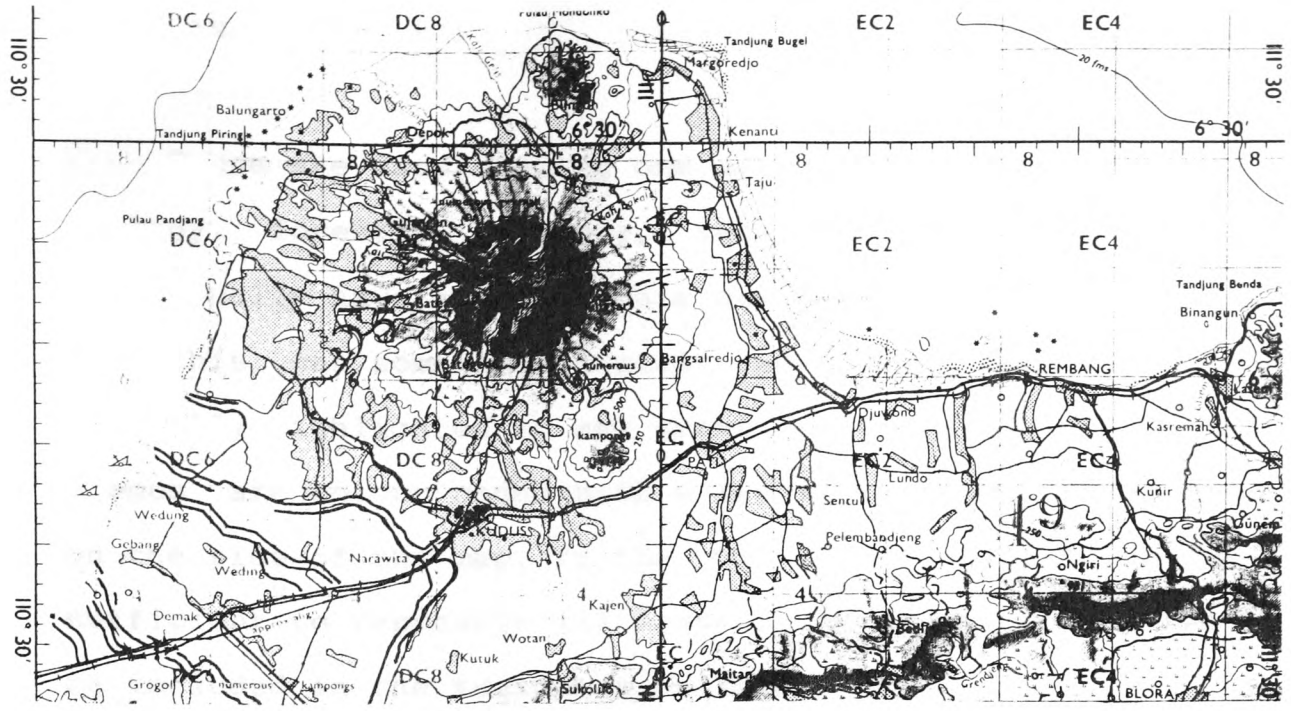


Fig 2.9 Aeronautical Chart TPC M-11D6, Ministry of Defence, 1971, Indonesia (Scale 1:500000).

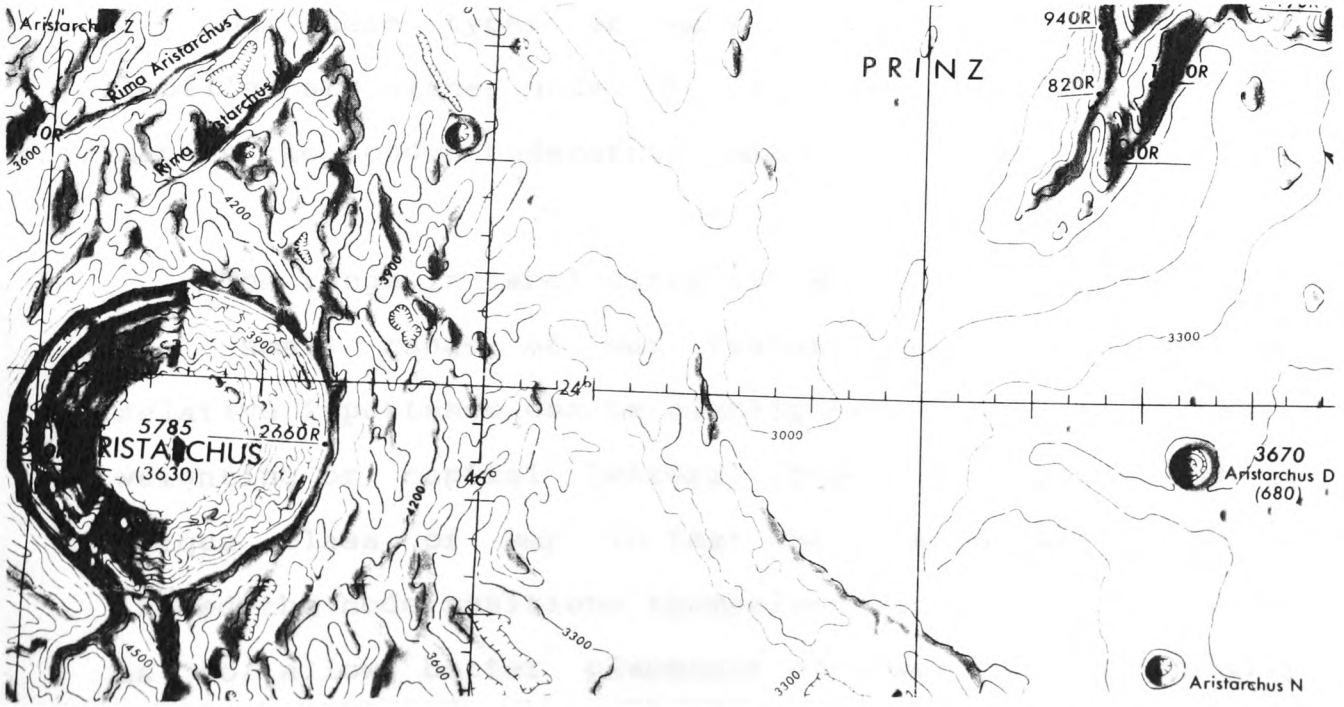


Fig 2.10 An unusual photomap: Lunar Chart of Aristarchus, LAC 39, United States Air Force, Nov 1963 (Scale 1:1000000).

2.1.9 ANALYTICAL MAPS

These are maps showing one particular theme such as geology, forestry, ecology, vegetation or soil surveys. They usually have a topographic map forming the base map, and this can sometimes contain labels, but these are not generally highlighted to any great extent. Most of the themes are colour coded with a description given in a key on the side of the map. If the colour coding scheme is not sufficient to represent all themes, then code labels may be included in the regions concerned.

2.1.10 NETWORK OR TOPOLOGICAL MAPS

On these types of maps, the principle features labelled are either nodes or links. An example of such a map is the London Underground map.

The range in label sizes is usually kept small due to the small range of map features presented, however relative importance can be highlighted by the use of heavy weighting or capital letters. Topological maps are a unique class of map in that the links between nodes or indeed the node positions themselves can be rearranged so as to allow better placement of names. This is usually allowed for during the map design stage, and so should not need to occur a second time.

2.1.11 SECTION SUMMARY

This section of the chapter has shown a variety of names and their usage as presented on different maps. The reader should now be gaining an appreciation of the necessary versatility required for a practical automated name placement system if it is to be designed to handle any type of map. The next section will present an example selection of name placement rules which will further enhance the readers appreciation of the complexity involved in annotating maps.

2.2 NAME PLACEMENT RULES

2.2.1 INTRODUCTION

There are three types of named map features, namely points, lines and areas, and each of these has its own individual naming protocol which varies according to the nature of the feature and the requirements of cartographic organisations. This section of the chapter will describe some example name placement rules, which illustrate the variety of rules and the information required in order to apply them. Each rule will be indexed by a rule number, quoted in square brackets.

One of the most authoritative works on cartographic name placement is the paper "Positioning Names on Maps" (Imhof). In it, Imhof postulates that each name has only one optimum position on a map. Generally this statement may be true, but there are examples, such as the 1984 edition of the Ordnance Survey Route Planner map (Fig 2.20), where feature and label density can be high enough to make it impossible to decide upon just one optimum position for a name.

2.2.2 GENERAL NAME PLACEMENT RULES

[2.1] Names should not overlap other names or point features (Freeman and Ahn, 1984). However on the Los Angeles Street map overlapping sometimes occurs between relatively low importance light coloured address number labels and more important dark coloured names (Fig 2.3).

[2.2] Names should help to reveal spatial and territorial extent (Fig 2.6) as well as importance, connectivity and differentiation between features (Imhof).

[2.3] Clear graphic association between the name and its object must exist (Fig 2.11). This determines style, size, size-gradation and quantity of type as well as map content. Hence, narrow-running types are preferred on small scale maps which have a higher label density than large scale maps (Imhof).

[2.4] Names should be easily read, discriminated and locatable despite being placed on a dense graphic background. Their legibility depends upon type form, size, colour and the position and arrangement of other names on the map as well as the general map contents (Imhof). Keates adds that names should avoid being placed over marked change in background colours.

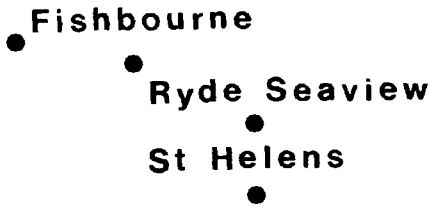


Fig 2.11 An example of poor graphic association and alignment between labels and objects. It is not very clear which labels refer to which points and whether "Ryde" and "Seaview" are separate labels [2.3].

Fig 2.12 Map Margin characteristic [2.14] (Imhof).

[2.5] Names should avoid overlapping underlying map features. However, if names have to overlap map features, it is preferable that they either cause the minimum amount of interruption (Keates) or overlap features of relatively low priority.

[2.6] Names cause less disruption with respect to underlying lines if they run across them at right angles (Keates).

[2.7] Type should stand as upright as possible and should not be inverted (Imhof).

[2.8] Diagonal labels should be written from lower left to upper right or from upper left to lower right (Imhof).

[2.9] Names should neither be evenly dispersed nor densely clustered (Imhof).

[2.10] In feature dense regions of a map, it is sometimes advantageous to split a multi-worded name into two or more lines if this improves the chances of finding a suitable position for it (Fig 2.2) (O.S. 4413, see O.S. Rules reference).

[2.11] On 1:50000 scale Ordnance Survey maps, names consisting of two or three long words should be written in two lines and justified towards the feature if possible ("Rhiwbina Fm", just north of "Tongwynlais" on Fig 2.4) (O.S. 2432j). However on larger scale Ordnance Survey maps, central justification is preferred (O.S. 4413).

[2.12] On 1:50000 scale Ordnance Survey maps, if hyphenated labels are split, then the hyphen should appear on the line where the split occurs ("Parc-y-Justice", south west corner of Fig 2.4) (O.S. 2439).

[2.13] Avoid placing two names of similar size and type adjacent to each other as this can cause visual confusion (Fig 2.11) (Imhof).

[2.14] On some topographic map sheets, names on the margin are placed as if adjacent sheets were joined together. Part of the word falling on an adjacent sheet is written in the empty margin usually with closely spaced and condensed letters (Fig 2.12) (Imhof). On 1:50000 scale Ordnance Survey maps, border names are at least one point size smaller than elsewhere on the map and have a minimum point size of six (O.S. 2437).

[2.15] Margins are kept fairly free of labels on atlas maps, especially in the vicinity of centre folds (Imhof).

[2.16] Word and letter spacing should remain fixed for a given label (Imhof). However in the case of atlas maps, this rule is sometimes broken ("YUKON TERRITORY" in Fig 2.6).

[2.17] On polar type maps, names can be written horizontally with respect to the bottom of the map. However, a preferable solution is to have a combination of some horizontal names near the polar zone (within two to three cm), and others lying parallel to lines of equal latitude, but not upside down on the upper half of the map (Fig 2.13) (Imhof).

2.2.3 POINT NAMES

[2.18] Labels above points are preferred to labels below since there is more chance of having ascenders than descenders (Imhof). There is also a slight preference amongst cartographers for placement of labels to the right and slightly above a point, this is because if both lie on the same horizontal, the alignment degrades legibility slightly. Names to the left should be avoided if possible due to a slight decrease in legibility and also because of the difficulty involved in right justified cartographic

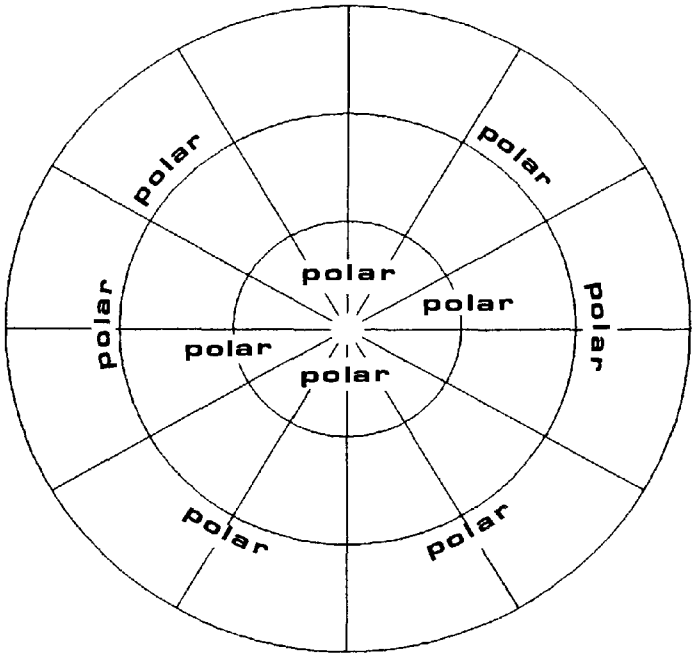


Fig 2.13 A representation of labels on Polar type maps [2.17] (Imhof).

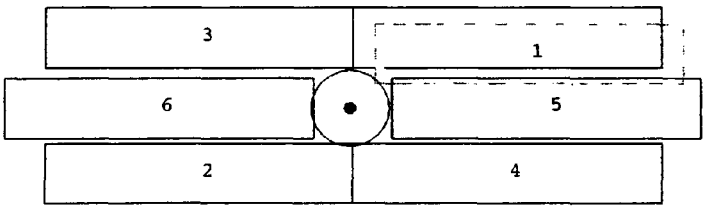


Fig 2.14 Preferred order of point label placement [2.18] (Freeman, 1985).

execution. However, Keates says that names sometimes have to be placed on the left of a point, and if this is the case then it is acceptable as long as the name is short. But in the case of regions of high name density, this can lead to confusion as to which point a name refers to (ibid). Fig 2.14 illustrates a preferred order of positions for a point label.

[2.19] On large scale maps point labels should be parallel to the grid lines and on small scale maps, to the lines of latitude (Fig 2.6) (Imhof).

[2.20] According to Imhof, names should not be placed on top of grid lines and from the diagram in his paper this seems to apply only to grid lines parallel with the labels concerned. This rule also appears to be dependent upon the size of the label, for instance it seems to be more strictly applied to small labels than to large labels.

[2.21] How close should a label be to its object? This depends upon the nature of a map, its scale and the size of objects and type. Typically for a smaller scale map, the separation distance is smaller (Imhof).

[2.22] Coastal place names should be written offshore for improved legibility (Imhof). This is especially true for small scale feature dense maps.

[2.23] When difficulty is encountered in placing a long point label horizontally on the coast, it is sometimes permitted to curve it towards the seaward side of the coast (Fig 2.6) (Keates).

[2.24] Sometimes summits of mountains have their labels written in arc form around them, but it is not recommended by Imhof because it makes the name slightly illegible. Therefore the use of horizontal labels on the upper right of mountain summits is recommended if possible.

[2.25] If both a name and a spot height refer to the same point, two labels exist for one feature. In such cases, it is usual to place the name far enough away so that the spot height is clearly nearer the point and preferably on the other side if possible (Imhof).

[2.26] On Ordnance Survey 1:50000 scale maps, spot heights should be placed along one side of a stretch of road (Anon, O.S.).

[2.27] For point names placed on or near to linear features such as a river or a boundary, place the label on the side of the line corresponding to the point. For point labels which are likely to span across a line, either split the name across the line or place it on the right hand side (Imhof).

[2.28] A special class of point labels are those which are placed at regular intervals along lines, such as grids on aeronautical maps. On such charts, latitude and longitude markers are labelled across the map surface. However a slight offset of these labels can occur to avoid overlap with underlying features and labels (The longitude values on the bottom of Fig 2.9).

2.2.4 LINE NAMES

[2.29] A linear name placed along a line should neither cling to it nor be too far away and should certainly not be written across it (Imhof). However, there appear to be several maps such as the Ordnance Survey Route Planner map, where this is contradicted (Fig 2.20).

[2.30] Low curvature sections of a line are preferable for placing a name on, especially near horizontal sections (ibid).

[2.31] Linear names should have either closely spaced letters or have very slightly spaced out letters. On 1:1250 and 1:2500 scale maps, linear features exceeding 150 and 300m respectively will have spaced labels (O.S. 4022).

[2.32] Linear names should be repeated at suitable intervals, particularly with respect to rivers and especially at tributaries (Imhof). On Ordnance Survey 1:50000 scale maps, the approximate repeat distance between road labels is 12.5cm (Anon, O.S.).

[2.33] On the Ordnance Survey Route Planner map, B class road labels are not usually permitted inside built up areas except if the B class road starts and ends in the region. The same applies to A class road labels except that these are sometimes found just inside built up areas (Fig 2.20).

[2.34] In cases where there are a lot of vertical lines, try and avoid frequent changes in reading directions of the labels from top to bottom and vice versa (Keates).

[2.35] Line labels should be written above lines if possible, because such labels usually have fewer descenders than ascenders (Imhof).

[2.36] On the Route planner map, road labels can be offset to avoid obscuring parallel features such as rivers and coastline.

[2.37] Vertical line names should be written according to rule [2.8] upwards on the left hand side of a map and downwards on the right hand half of the map. Both cases are valid in the middle of a map (Imhof).

[2.38] Adjacent curved line names must always run in the same direction. Also, one should try and make the labels run from left to right and have the letters the correct way up (ibid).

[2.39] On 1:50000 scale Ordnance Survey maps, the order of labels on consecutive contours are written uphill (Anon, O.S.), thus it is possible for contour numbers on the north sides of high relief regions to be written upside down thus contradicting rule [2.7].

[2.40] Contour labels are often represented in a ladder form either straight, slanting or curved, but should not be closer than 5mm to each other on a 1:50000 scale map (Anon, O.S.).

[2.41] Contour labels have a much lower importance than other map features and can often be found shifted out of the way of other feature names. Sometimes, they can be repeated at intervals along contours.

2.2.5 AREA NAMES

These refer to areal objects such as fields, mountains, woods, marshes, sands, ridges, hills, counties, districts, and islands. Some areas have boundaries such as lakes and counties, whereas others which include mountain ranges, geographical areas, and seas are unbounded.

[2.42] The extent and positions of the area name should be central and reflect the true extent of the area. On large scale maps, names can often be placed inside and sometimes next to buildings (Imhof) (Fig 2.1).

[2.43] Small areas which are too small to place a name in, are treated as point like features during labelling. At small scales, small island and island group names sometimes have this characteristic (Imhof)

[2.44] Where a named area is unbounded, the limits should be approximately defined by the extent of the label (Keates). This applies to large physical features such as mountain ranges and deserts.

[2.45] There must be clear areal association between label and feature, therefore area labels should be bent or stretched as much as possible so as to spread out across the horizontal axis of the area (Imhof).

[2.46] Names should extend along the longer axis of an area and its ends should lie within one and a half letter intervals of the area edges (Keates).

[2.47] When areas overlap, the less important area labels are usually moved out of the way, since they have lower priority.

[2.48] Areal river labels should gradually widen towards river mouths (O.S. 2434c).

[2.49] On the Ordnance Survey Route Planner map, there appears to be a slight tendency for unbounded sea and bay area labels to be placed horizontally.

[2.50] In the case of area names such as geographical regions greater than 50km sq in area on Ordnance Survey 1:2500 scale maps, these are only shown marginally, and are not included on 1:1250 scale maps (O.S. 4145).

[2.51] On Ordnance Survey 1:50000 scale maps, large geographical feature labels, such as mountain ranges, can run across several sheets (O.S. 2434b).

[2.52] Elongated area features usually have label letters spaced out. However this should be avoided for short labels, since they become difficult to read. Instead, a short label can either be repeated at regular intervals along the area or can be written using large, bold type (Keates).

[2.53] For linear area names, try and place names on a section which is as near to the horizontal as possible and try to avoid sharp sections (Imhof).

[2.54] Area names are preferred if they curve towards the horizontal rather than away (Imhof).

[2.55] Tilted area names should always be noticeably curved (Imhof). However this depends upon the map style.

[2.56] Long area names can be doubly curved but should avoid being placed on uneven or sharp curves (Imhof).

[2.57] Avoid curving area labels through more than sixty degrees (Imhof).

[2.58] Small area labels should be moved away from the alignment of larger ones (Keates).

[2.59] One should avoid the aesthetic clash of having adjacent parallel area features curving in opposite directions (Freeman).

[2.60] Letter spacing should be constant for a label and ideally be no more than twice as wide as letter height. The extent of the spacing will depend upon map content and feature density (Imhof).

[2.61] For town plan maps, care is required in order to space out either the letters or the words making up a street name, so that a label can represent the whole length of the road it relates to. In the case of very short roads, letter size and spacing is reduced in order to allow the label to fit into the shortened area available (Fig 2.2).

[2.62] On town or city street maps, abbreviations for named features such as streets, roads, terraces, squares, and crescents are sometimes used where the label has to fit into a restricted space (Fig 2.2).

[2.63] On town or city street maps, in very confined circumstances, road names can be placed partly and sometimes totally outside their casing ("FITZHAMON LA.", north west of "CARDIFF CENTRAL" in Fig 2.2).

[2.64] On town and city street maps, area place names have priority over road names since if they lie over roads, the associated road names have to be either shortened, shrunk in size or split to avoid overlap (Fig 2.3).

[2.65] On the yachting chart (IMRAY C10, Western English Channel, April 1983), scaled at 10 nautical miles to the inch, geographical sea area labels tend not to be expanded to show the areal extent.

[2.66] Area labels corresponding to county and national regions on the yachting chart are placed horizontally and centrally inland usually between 2 to 5cm from the coast.

[2.67] There is an unusual example of a split area label in the case of a "Submarine Exercise Area" label just below Lyme Bay on the yachting chart (Fig 2.15). The label is split into three horizontal words, but the placement of each word is in a staircase arrangement.

[2.68] Another interesting example on the same yachting chart is the "Inshore Traffic Zone", near the Scilly Isles. This particular label is split into three lines, but the spacing between lines is not equal. This appears to be in order to avoid overlapping a sea depth sounding label (Fig 2.8).

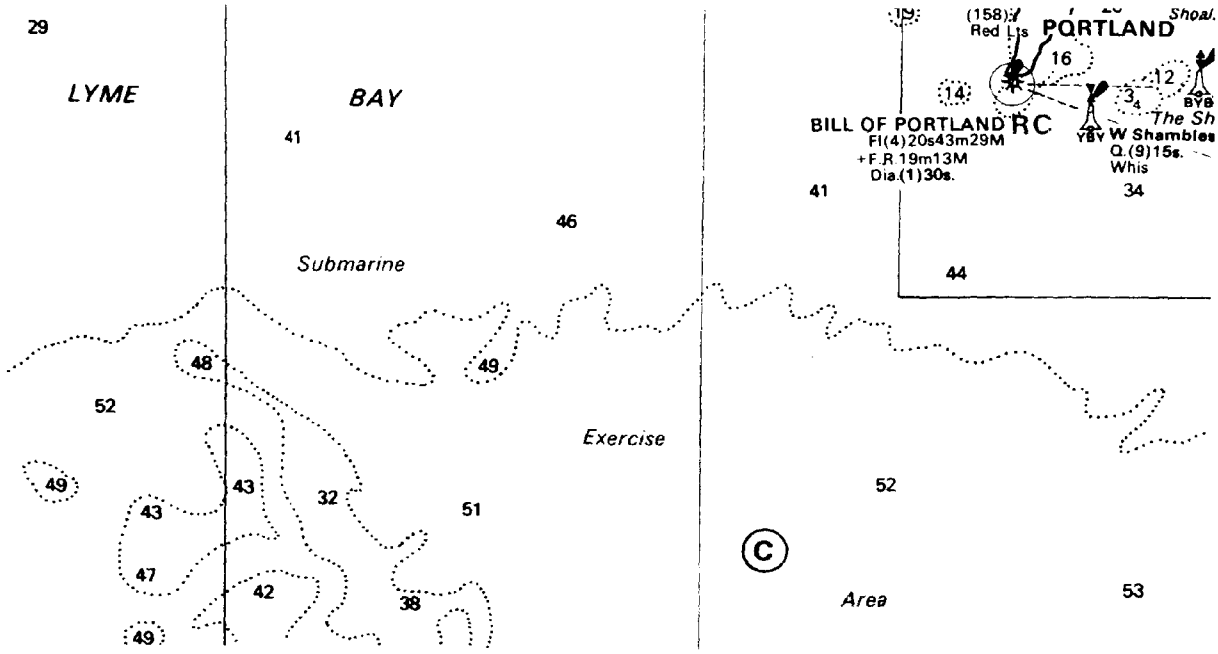


Fig 2.15 A "Submarine Exercise Area" label from the Imray C10 Yachting Chart of the Western English Channel, April 1983 [2.67].

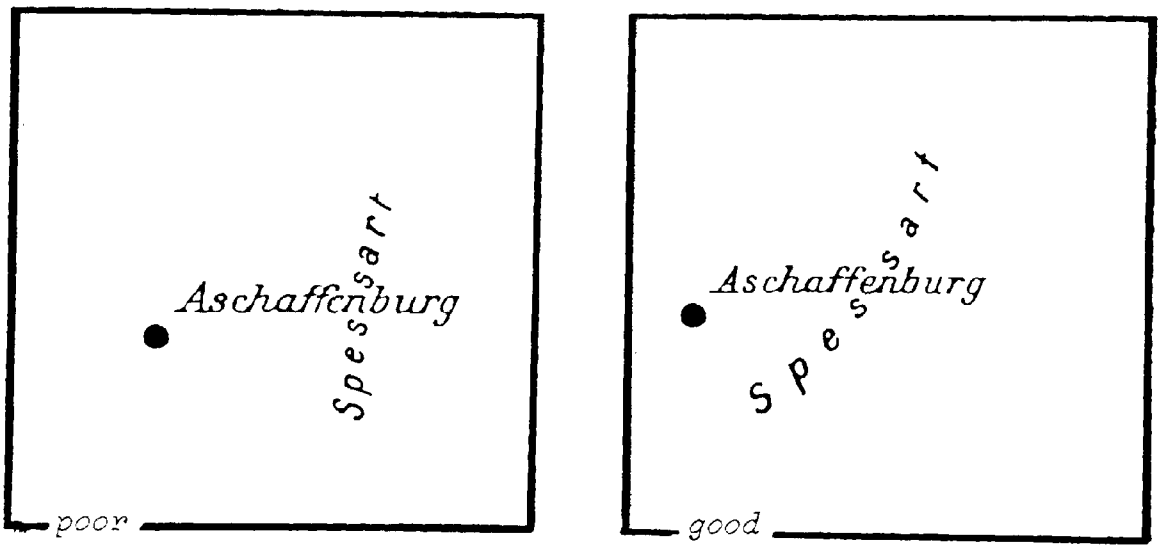


Fig 2.16 Names should not cross each other perpendicularly [2.75] (Imhof).

[2.69] On administrative area maps, it is sometimes permitted to allow some area names to flow into the sea ("SOUTHEND-ON-SEA", Fig 2.7).

[2.70] To help identify very small regions on an administrative area map, numbers are sometimes used in preference to names and a key is provided to establish the identity of the areas (Fig 2.7).

[2.71] Area names should be placed so that they lie on the perceived centre of gravity of the area (Anon, O.S.).

[2.72] An area name is ideally placed when the regions between the area boundary and the left, right, upper and lower edges of the label are equal (King).

2.2.6 MIXING OF DIFFERENT LABEL TYPES

Most maps contain all three types of label, therefore care must be taken in the mixing and presentation of these. Keates reminds us that the placement of one name by itself is not nearly as difficult as placing all names simultaneously.

[2.73] One should attempt to position names which have to be spaced out to represent large features first, since these are the most difficult to place. This should then be followed by smaller point and line labels (Keates).

[2.74] Both Imhof and Keates say that labelling in high density areas should occur in an outward direction, towards areas of lower density. By shifting labels on the outskirts of such regions away from the centre, this frees potentially occupied space for labels inside and thus improves the chances of finding optimum label positions for these labels.

[2.75] In cases where labels of different sizes all share the same region of the map, try to avoid small names intervening between big letters belonging to large extended labels. If this is not possible then avoid placing small names perpendicularly across large spaced out names (Fig 2.16) (Imhof).

[2.76] On some atlas maps of countries, it appears to be permissible for smaller labels to nearly touch the edges of letters making up administrative or geographical region labels. However, the size difference between the labels is usually quite considerable ("Hall I." below the "r" in the "Bering Sea" Fig 2.6).

2.2.7 INFORMATION NEEDED

Mentioning every single piece of information required for the application of all of these rules is not the purpose of this section of the chapter. However, the information described below has been selected to show the elaborate knowledge required to apply just a few of these rules.

The application of most of the rules requires the ability to detect overlaps between labels and also to be able to determine what underlying features are present. Rules [2.1], [2.4] and [2.5] require not only the ability to find out what underlying features are, but also their type, importance and colour. Even the orientation of underlying line features is required for rule [2.6].

Several of the rules require the spatial extent of named features [2.2], their orientations [2.45], curvature [2.53], area and width [2.48] to be known. Unbounded area features such as river mouths must have the approximate limits of the feature defined [2.44].

Global map information such as feature density [2.9], the distribution of labels and features [2.10], grid line locations, margin locations [2.14], [2.15] and which half of a map a name lies on is also needed [2.17], [2.37].

The placement of many labels depends upon the location of other labels, for instance the placement of a contour label depends upon the position of other adjacent contour labels along a slope [2.40]. Road labels are often repeated, and the position of each occurrence must be known to prevent the placing of two identical road labels too close together or too far apart [2.32].

In the case of multi-worded labels, the contents of the label has to be known in order to decide whether and, if so, where to split the name [2.10], [2.11], [2.12]. Letter spacing and size [2.13], [2.14], [2.16] must be available, as well as orientation, for names in general [2.7], [2.8].

Labels near dividing lines such as boundaries [2.27], roads [2.26] and the coast [2.22], [2.23] often require knowledge about which side of the line they are to lie on.

The reader is invited to re-read some of the above rules and ask what further basic information is needed to utilise them. The rules discussed are only a small sub-set of those actually used. An automated name placement system which is capable of placing names on any type of map must have access to such an extensive range of information.

2.2.8 SECTION REMARKS

It is of interest to note that according to Imhof, name placement rules tend not to be totally exclusive. For instance, there are often exceptions and ambiguities over which rules to apply under certain situations. There are also contradictions, as when Freeman and Ahn (1983) state exactly the opposite to rule [2.54].

This chapter has so far described different types of names and given a range of name placement rules derived from the literature and by asking experts. It has also mentioned some of the information needed to apply the rules. The rules given are only a very small selection of those actually used. The next section will describe ways of extracting unwritten rules and verifying the use of name placement rules by analysing labels statistically.

2.3 ANALYSIS OF MANUAL NAME PLACEMENT

2.3.1 INTRODUCTION

When extracting rules from cartographers or reading about rules in the literature, one essentially has to rely upon the information provided as being both accurate and complete. In fact, the majority of name placement rules given in the last section are incomplete since they do not mention frequency of use on different types of map. Also, some of the more general rules may not be suitable for certain types of maps.

One method for checking name placement rules is to analyse statistically the positions of names on maps and their configurations. When checking on point, line and area name usage, it is useful to classify them further into uniquely defined positions or configurations with respect to their associated features. Using these classified configurations, a statistical analysis can then be made of their frequency of use. From this, one can gain an idea of their relative importance and perhaps even discover some unfamiliar rules.

Four 100x100km library grid square regions (Fig 2.17) of the 1984 edition of the Ordnance Survey Route Planner map were selected for the analysis of the use of different types of name placement rules. Library grid squares 201

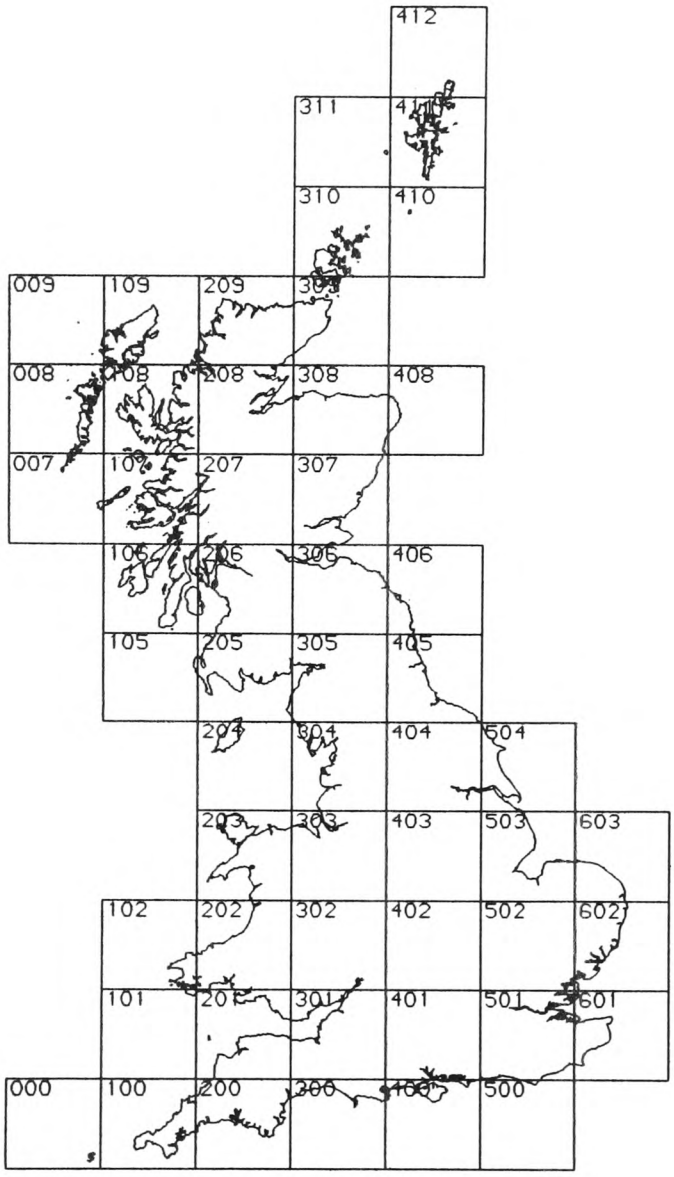


Fig 2.17 Ordnance Survey Route Planner map 100x100km library square distribution over Great Britain.

(Fig 2.18) and 301 (Fig 2.19) covering Devon, Somerset and the South Wales region were selected because it was of interest to compare the statistics of two adjacent regions and also to see what effect extensive coastline had on statistics. Library square 501 (Fig 2.20) was selected to test how name placement was affected by high feature density regions such as London. Finally, library square 208 (Fig 2.21) of the Inverness region was chosen to study how name placement is influenced by areas of low feature density.

In the case of a point label, there are an infinite number of positions that the label can occupy, even if the label is restricted to being a fixed distance away from the point and is horizontal. Therefore for statistical purposes the label positions must be classified into a finite number. The positions must be selected so that the separation distance between adjacent label positions is small enough to approximate to a continuous distribution of positions and large enough to allow for a significant statistical sample in each position. Fig 2.22 illustrates the twenty positions adopted. The label positions selected are at principal locations covering horizontal, vertical and diagonal offsets with respect to the point. Table 1 (Appendix 1) presents the frequency of occurrence of label configurations around settlement point features in the chosen library grid squares.

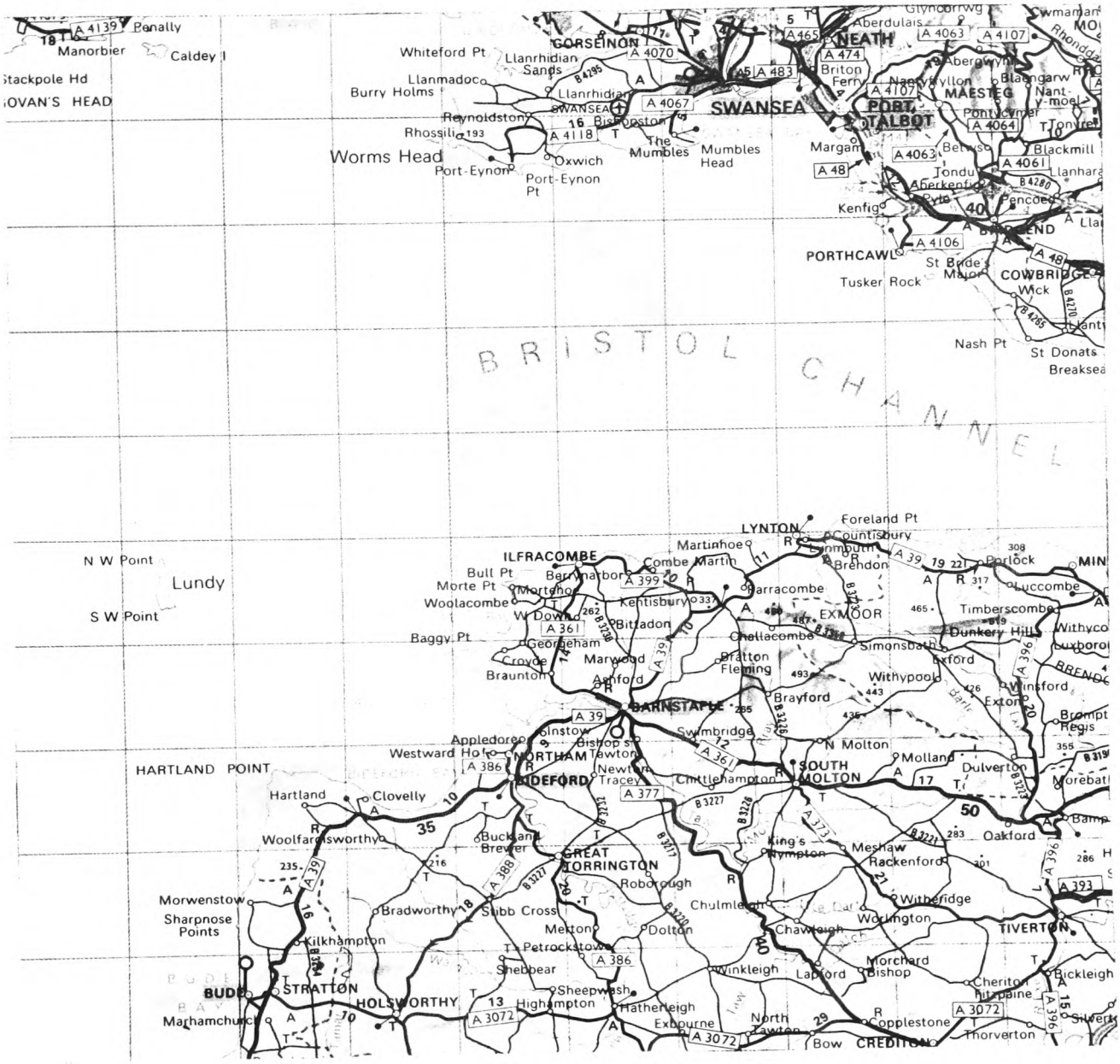


Fig 2.18 Ordnance Survey Route Planner Map, Library Square 201, North Devon (Scale 1:625000).

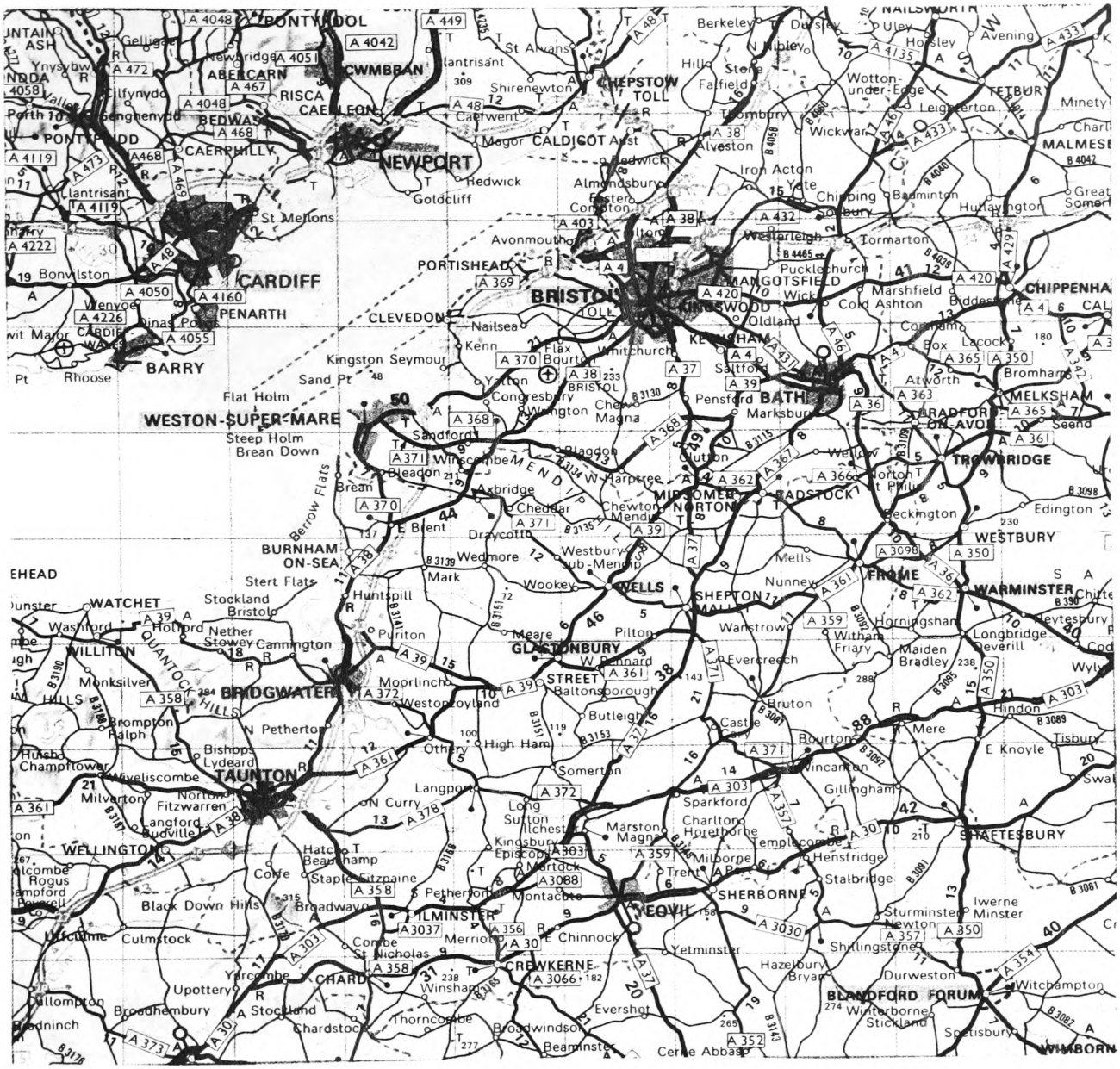


Fig 2.19 Ordnance Survey Route Planner Map, Library Square 301, Somerset and South Wales (Scale 1:625000).

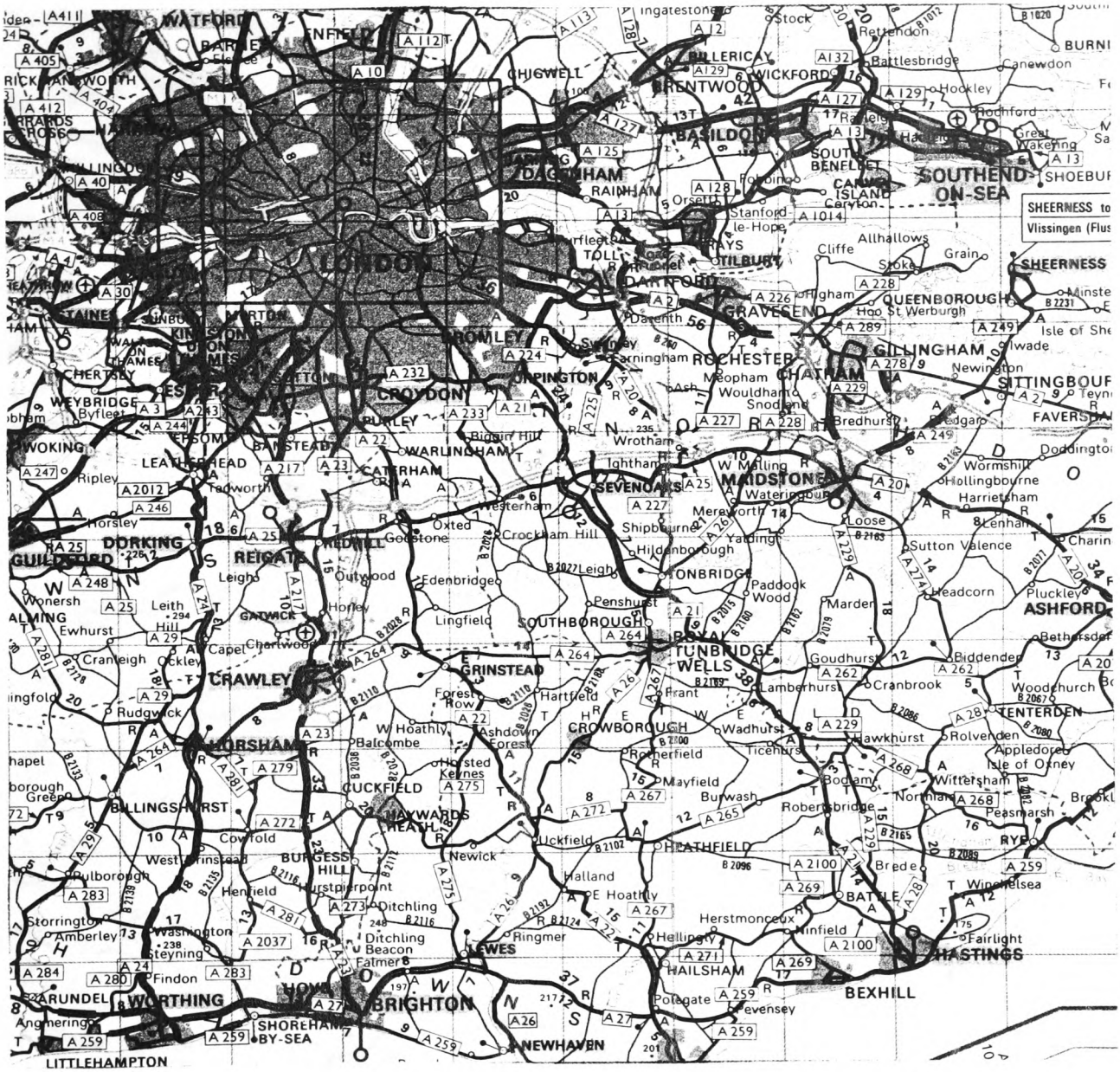


Fig 2.20 Ordnance Survey Route Planner Map, Library Square 501, London (Scale 1:625000).

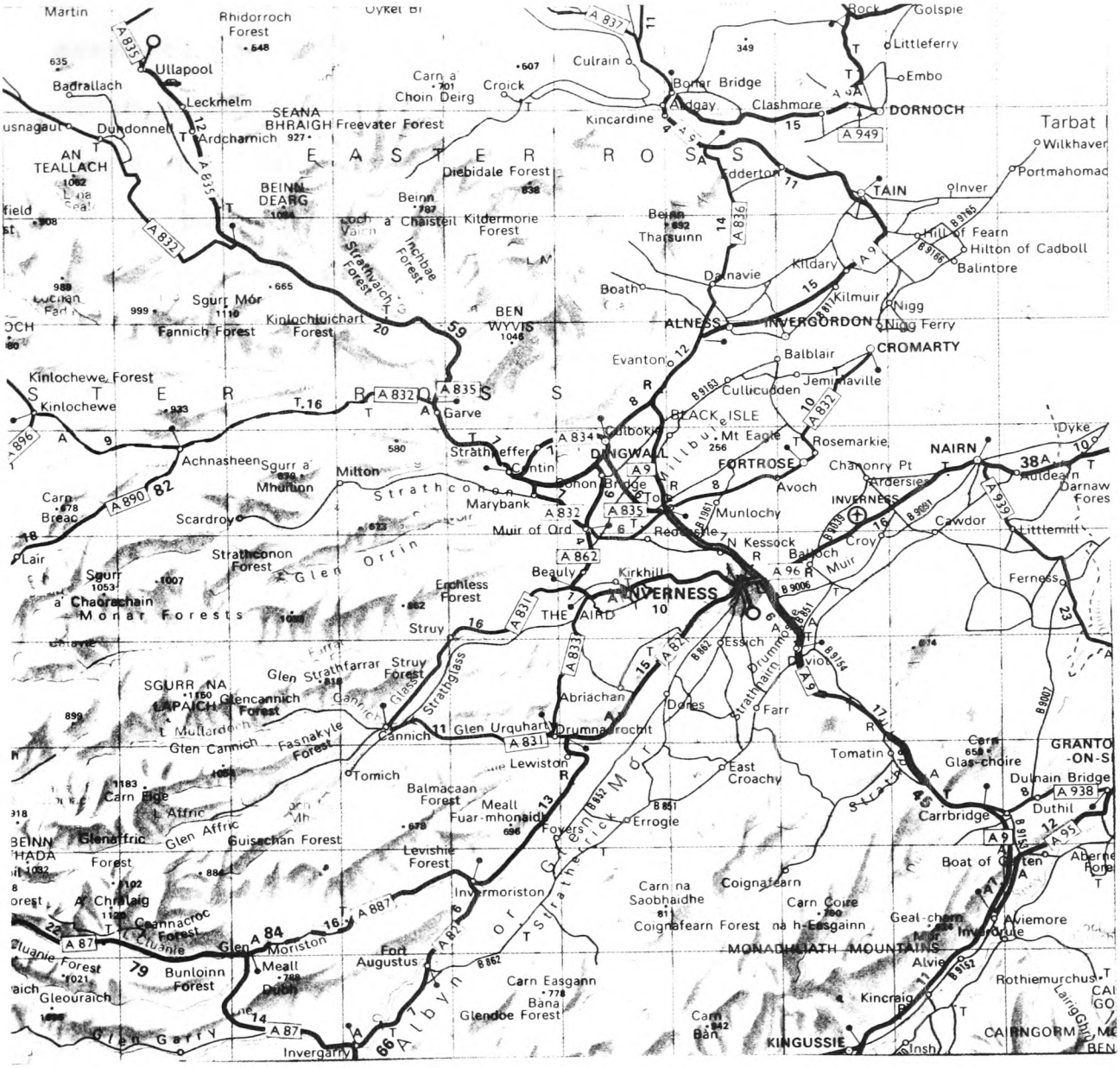


Fig 2.21 Ordnance Survey Route Planner Map, Library Square 208, Inverness (Scale 1:625000).

In the case of line names on the Route Planner map, the position of a label on a line is more difficult to measure. In fact, the position of a label along a line is often immaterial as long as it lies in regions of free space. A matter of greater importance is how the line label should be configured. Should it lie along the line, above it, below it, horizontally across it, a combination of these or even arrowed? In the case of the Route planner map, 11 possible configurations were considered and these are illustrated in Fig 2.23. The frequency of use of label configurations of three different classes of line features was analysed individually for each library grid square. The line features selected included A class roads (Appendix 1, table 2), B class roads (Appendix 1, table 4), and rivers (Appendix 1, table 6). Because of the low number of motorway labels, it was decided not to perform a statistical analysis of their label configurations. In the case of B class roads and rivers, only four discernible configurations existed and these are illustrated in Fig 2.24.

According to the rule [2.30], the preferred angle of placement for a line label is a horizontal. Does the same apply to line names on the Route Planner map? By studying the frequency of occurrence at different angles of tilt of road and river names over a range of intervals of twenty degrees, some indication was obtained (Tables 2.3, 2.5 and 2.7).

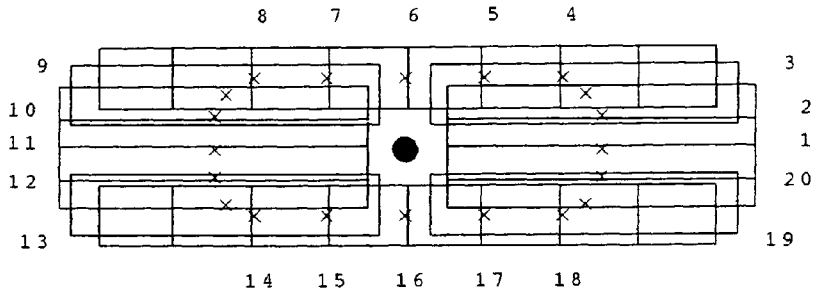


Fig 2.22 Point label positions (Positions refer to the centre of label, marked by a cross)

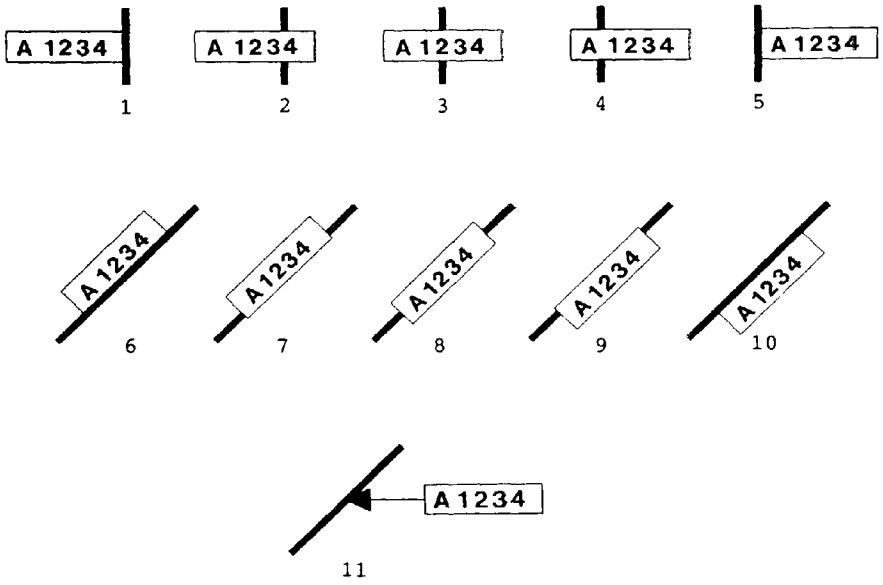


Fig 2.23 A class road label configurations.

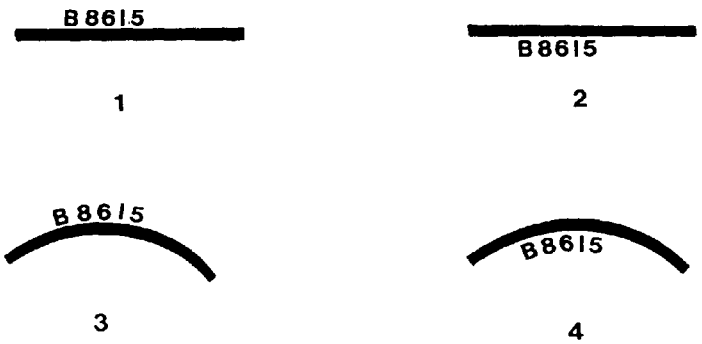


Fig 2.24 B class road and river label configurations.

Rule [2.32] refers to the average repeat distance of road labels on 1:50000 scale Ordnance Survey maps as being 12.5cm. On the 1:625000 scale Route Planner map, one might naively expect it to be a scaled distance of 1cm. The typical repeat distance of road labels was determined by averaging over several repeat distances. The range of road repeat distances was found in a similar way. However, because of the wiggliness of many of the roads at 1:625000 scale, it was only practical to measure straight line distances.

Area labels were also studied, this was of particular relevance due to the large number of area rules given in section 2.2. Unfortunately, on the Route Planner map, area labels were too few in number and too many in different areal feature classes, to investigate statistically. Instead, it was decided to analyse the configuration of area labels on the Ordnance Survey 1:625000 scale Administrative area map of Great Britain which uses the Route Planner map as its base map. On the Administrative area map, the following areal configurations were used: horizontal, diagonal, curved, marginal, numbered and arrowed (Fig 2.25). Furthermore the area labels could also be split.

Finally, point settlement labels were investigated to see under what conditions they were split.

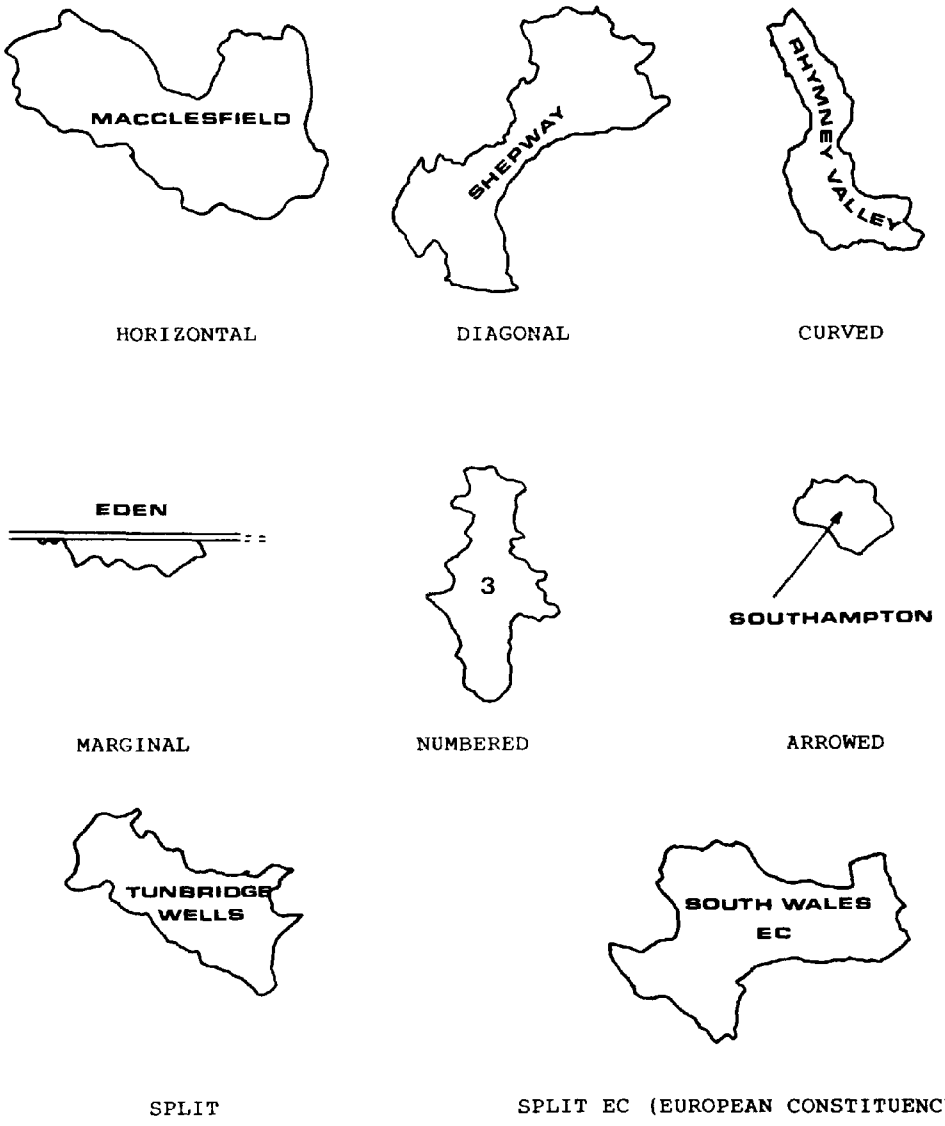


Fig 2.25 Administrative area label configurations.

2.3.2 RESULTS

For point labels (Appendix 1, table 1) over the four regions covered, on average, 15% of labels were placed directly to the east of the point (1) and a similar number to the east and slightly below the point (20). The next most preferred position appeared to be directly to the west of the point (11) which is contrary to rule [2.18]. This was then followed in order of preference by a position to the east and slightly above the point (2) and another directly below the point (16). The least preferred positions appeared to be the intermediate vertical displacement positions (5, 7, 15 and 17) and diagonal positions above the point (3 and 9).

On average, A class road labels were placed with equal frequency diagonally along the road and horizontally across it. There appears to be an exception however in regions of low label density, where approximately three quarters of the labels were placed diagonally along the road.

The results show that over the four regions covered, approximately two thirds of A class road labels were usually placed horizontally (Appendix 1, table 3). However, there was a suggestion that horizontal placement of such labels was preferred more in regions of high label density (London 69%) than in regions of low label

density (Inverness 45%). The remaining range of angles seemed to have no preferred orientation, except that in the Inverness region, the sum of the frequency of occurrence of angles between +/-30 to 70 degrees seemed equal to the occurrence of horizontally placed labels.

It was found that straight B class road labels were three times more likely to be used than curved labels (Appendix 1, table 4). This may show some indication of the preference of labelling straight sections of linear features. The angles of B class road labels appeared to have a slight preference for horizontal placement, but otherwise no obvious preferences appeared to exist (Appendix 1, table 5).

The placement of river labels showed no obvious tendencies towards preferred angles (Appendix 1, table 7), however there was a slight preference for the use of straight labels (Appendix 1, table 6).

The average repeat distance for A class road labels was found to be 30 +/- 12mm and ranged from 6mm to 63mm. For B class road labels, this was 27 +/- 12mm, covering a range from 15mm to 52mm. The average repeat distance for motorways appeared to be somewhat greater at 47 +/- 15mm and ranged from a minimum of 30mm to 74mm. The average repeat distance for line labels may be related to the frequency of occurrence of its feature class, the greater

the number of line features of a particular class, the shorter the average repeat distance.

On the administrative area map, the European Constituency areas were distinctly different to county and district areas, in that nearly all of the labels were horizontal, and all had the abbreviation EC suffixed to them. 80 to 90% of both district and county labels were horizontally placed and approximately 10% were both horizontally placed and split. Curved labels were more preferred for county areas than for district areas

Finally, with regard to the splitting of point names, by studying the Route Planner map, the following rules were deduced which are additional to those given in 2.2.3:

[2.77] On the Route Planner map, multi-worded labels are very rarely split when they are less than 10 letters in length.

[2.78] On the Route Planner map, on average three quarters of labels with 10 or more letters are usually split into two lines. However there are several examples of multi-worded labels in excess of 10 letters in length, which are not split if they lie in regions of very low feature density.

[2.79] On the Route Planner map, if a word in a multi-worded label is less than three letters in length and is a prefix to the next word, such as "St" in "St Arvans", then the label must not be split between the two words. Similarly, if an abbreviated word of less than three letters, such as "Pt" or "I" is found at the end of a label, a split must not occur at this abbreviation.

[2.80] On the Route Planner map, when labels are split, the split should be positioned so that the length of the split label is kept to a minimum, after allowing for rule [2.79].

2.3.3 SECTION DISCUSSION

For point label positions, the order of preference generally appeared to remain fairly constant in different library grid squares for the most and least preferred label positions. However the order of preference of the remaining positions tended to vary randomly between library squares, as one would expect, because no particular effort had specially been made either to use these positions or to avoid them.

The only noticeable anomaly amongst the results from the different library squares for point labels was in library square 508, where point label position 18 appeared

to be more frequently used than in the other squares. Also, in the same square, point labels were much less likely to be split, presumably due to the lower feature density [2.10].

There seemed to be indications that on label dense regions of the map, it was preferable to place A class road labels horizontally rather than diagonally since more labels could be fitted onto the map this way. However, B class road and river labels appeared to lack preferred angles, which may reflect their relatively low importance compared to other labels.

Neighbouring library grid squares 201 and 301 were similar in certain respects except that library square 301 had a much higher feature and label density. No noticeable influence of large amounts of coast were found in any of the squares. Possible anomalies between different library squares may be due to to several factors, for instance the global alignment of certain features or the possibility that the map may have been labelled by different cartographers using their own personal rules.

The analysis of area labels on the administrative area map showed that for this particular map, the majority of labels were horizontally placed and did not necessarily have to represent the full areal extent of the area concerned. This meant that rule [2.45] was not applicable

for this type of map.

The counting of the occurrence of different label configurations was found to be a very tedious task. Nevertheless the technique of statistically analysing name configurations proved capable of revealing the apparent, preferred use of different name placement positions and configurations. By relating this to other factors such as feature density on the map new rules can be revealed.

2.4 CHAPTER DISCUSSION

The first two sections of the chapter illustrated the extreme complexity involved in annotating maps that an ideal automated name placement system would have to cope with. In section 2.3 we saw how the range of label positions was restricted to a finite number for statistical purposes, and also, how label configurations could be classified. The same techniques are of relevance to the design of an automated name placement system in that reducing the number of label positions to a finite number reduces the complexity of the problem.

Also label configurations could be restricted to just those which are commonly used and those, which although rarely used, are essential. This restricted set of label configurations and positions has the effect of making the task of automated name placement easier but will reduce cartographic name placement freedom slightly. An example of a rare but essential label configuration is the use of numbered area labels on the Ordnance Survey Administrative area map (Fig 2.25). It could be argued that these numbers should be used in preference to arrowed labels on the map, which are also rare, thus reducing the range of configurations further. If the results are not satisfactory, then either further name placement rules and

configurations will need to be added or it may be decided to adopt a semi-automatic approach by editing some of the name placements manually.

The reader may have noticed in section 2.2 that several cartographic authorities have different opinions on which rules are permissible, also a few contradictions regarding rules were found in section 2.3. This clearly demonstrates the need to design a name placement system which is not restricted to just one cartographic application, but which can cope with any cartographic convention by allowing for new rules to be added.

CHAPTER 3

CARTOGRAPHIC DATA STRUCTURES

3.1 INTRODUCTION

This chapter discusses different cartographic data structures and selects which are suitable for automated name placement systems. Cartographic data is commonly produced in two forms: vector and raster. Vector data consists of three basic entity types: points, lines and areas. Raster data consists of a grid square representation of the map usually in the form of an array of integer numbers.

In order for a name placement system to function comparably to its human counterpart, it requires a suitable database with the following three specifications. Firstly, it must have access to structured cartographic data whereby human cartographic visual questions relevant to name placement can be answered relatively easily. Secondly, in order to be able to construct a practical working system, within hardware limitations, memory space must be used economically by utilising compaction techniques such as a run-length encoding. Thirdly, the data structure must not appreciably limit the speed at which cartographic information can be retrieved.

An example of a data structure where cartographic visual questions cannot be answered easily, without sophisticated interpretation routines, is "spaghetti vector" data (De Simone, 1986). This type of data is typically generated, when large scale maps are digitised into points, lines and areas, without attempting to structure the data into recognisable features. Often errors are present, such as lines at junctions which do not always exactly connect.

More advanced forms of cartographic data usually include attributes and relations between the features in the data. For example, in the case of the Ordnance Survey's 1:625000 Route Planner database, a feature serial number of 2789 might refer to some data, in the roads and settlements dataset 3, with an associated feature code of 42, inferring a "narrow main road" and a second attribute referring to the features name. The new Ordnance Survey digital 1:50000 scale data model, allows for additional flexibility by catering for dual meanings in vector data, such as a line which forms part of a river and also a boundary. Additional left/right pointers are used for boundary line data, so as to point to the two corresponding adjacent area features (Haywood, 1986).

However even feature coded vector data is not capable of supplying, rapidly enough, all the necessary information that a sophisticated name placement system

requires. For instance it requires information such as, "What proportion of a rectangular area occupied by a label is empty space on the map?", or "What classes of feature lie underneath the label?" (Rule [2.5]). Extracting such information using just vector data alone is a rather time consuming process which involves searching through several vector records to find data in the region concerned. Instead raster data can be used to yield this kind of regional information with relative ease. Unfortunately, raster data generally occupies more memory storage space than the equivalent in vector form.

This chapter will show that, whilst there are many advantages to using one type of data structure over another, both raster and vector cartographic data storage are needed in order to satisfy the general information requirements of a sophisticated name placement system. Also, much will depend upon how point, line, area and raster data are related to one another, and the use of attributes.

3.2 VECTOR DATA

There are several ways of representing cartographic data in a vector form. For instance the Ordnance Survey's 1:625000 database (Haywood, 1984) uses a data structure consisting of just points and lines. This has proved suitable for cartographic purposes involving the selection of different feature codes to plot and has been used by the Ordnance Survey in the production of the 1986 edition of the Route Planner map. Other approaches make use of polygon data structures such as POLYVRT (Peucker and Chrisman).

The function of a point cartographic feature is to represent a single discrete location on a map. The cartographic feature at the location specified is a symbol usually representing a physical feature, but can also indicate geographic, historical or political significance.

The purpose of line cartographic data is to describe linear entities such as rivers or roads, and also line features which are not plotted on the map, such as the path of a road through a tunnel. A named line feature can undergo changes of attribute along its length. For instance, on the Route Planner map, named roads can change from primary routes to dual carriage ways, to single carriage ways and so on. Such features consist of separate links, each of which can have a different feature code,

but all of which have the same object name pertaining to the line. If a record is kept of which links join at junctions, called nodes, then connectivity relationships can be found of one link to another across a large section of a map.

A link consists of a string of adjacent points whose attributes are all the same (Haywood, 1986). The coordinates of these points can either be "absolute" which specify the complete coordinates of each coordinate pair in the link, or "relative" which give the complete coordinate of the first point in the link, followed by offsets for each of the remaining coordinate pairs. The advantage of using relative coordinates is that, because the offset distance between adjacent coordinate pairs making up a link is small, less storage space is occupied. The disadvantage of relative coordinates is that one cannot predict what the maximum likely separation distance will be between adjacent coordinate pairs.

One means of getting around this problem would be to use two files, one to store the first initial absolute coordinate of each link, a pointer and the number of records to read from the second file. The second file would contain the relative coordinate pairs. A special number flag could be used in the second file to indicate that the next record must be read if a relative coordinate is encountered greater than the expected field width.

However, this introduces an extra data processing stage into the system.

Another way of reducing vector data storage requirements is to filter out redundant coordinate pairs which are not needed for name placement. This can be achieved because label placement positional errors smaller than a fifth of the smallest character height will probably not be noticeable for most labels. Thus any line characteristics smaller than this threshold can be removed and this will result in the number of link coordinate pairs being reduced by a significant factor of perhaps two or more. Several filtering algorithms exist which can be used for this purpose such as "Perpendicular distance", "Angular" (McMaster) and the "Douglas-Peucker" (Douglas and Peucker, 1973) algorithms.

Not all features are clearly defined by vector data, for instance, in the case of a bay, on the Ordnance Survey 1:625000 database, a location point, called a "seed", will be given to indicate approximately the centre of a bay. Visually the extent of a bay is usually defined by the inland coastline and a limiting line between the two headlands or points on either side of the bay. Unfortunately, computation of the seaward extension of a bay is rather difficult to perform using just coastal boundary data because although many clear cut cases of headlands can be found on either side of a bay by

searching for sharp angles in the coastline, such an algorithm would have to avoid accidentally interpreting small irregularities in the coastline as headland limits. For instance in Fig 2.18, does "BARNSTAPLE OR BIDEFORD BAY" extend from "HARTLAND POINT" to "Baggy Pt" or from "HARTLAND POINT" to "Morte Pt"?

Channels are another case of unbounded areas. Although they are defined on both sides by coastline, it is not a trivial matter to define the limits at either end of a channel, without some geographical knowledge of the channel itself. For instance in Fig 2.18, does the "BRISTOL CHANNEL" terminate on its west edge between "Stackpole Hd" and "HARTLAND POINT", or "Worms Head" and "HARTLAND POINT", or "Worms Head" and "ILFRACOMBE"?

Although hills are another class of undefined region, providing that contour data is available, then highland regions can be highlighted by thresholding out contour heights. Unfortunately, the extent of such highland areas is not always easy to judge from contour data alone (Fig 3.1).

Because of all the associated problems with unbounded area data, the only practical solution is to expect some approximate boundary data to be present in the data to begin with.

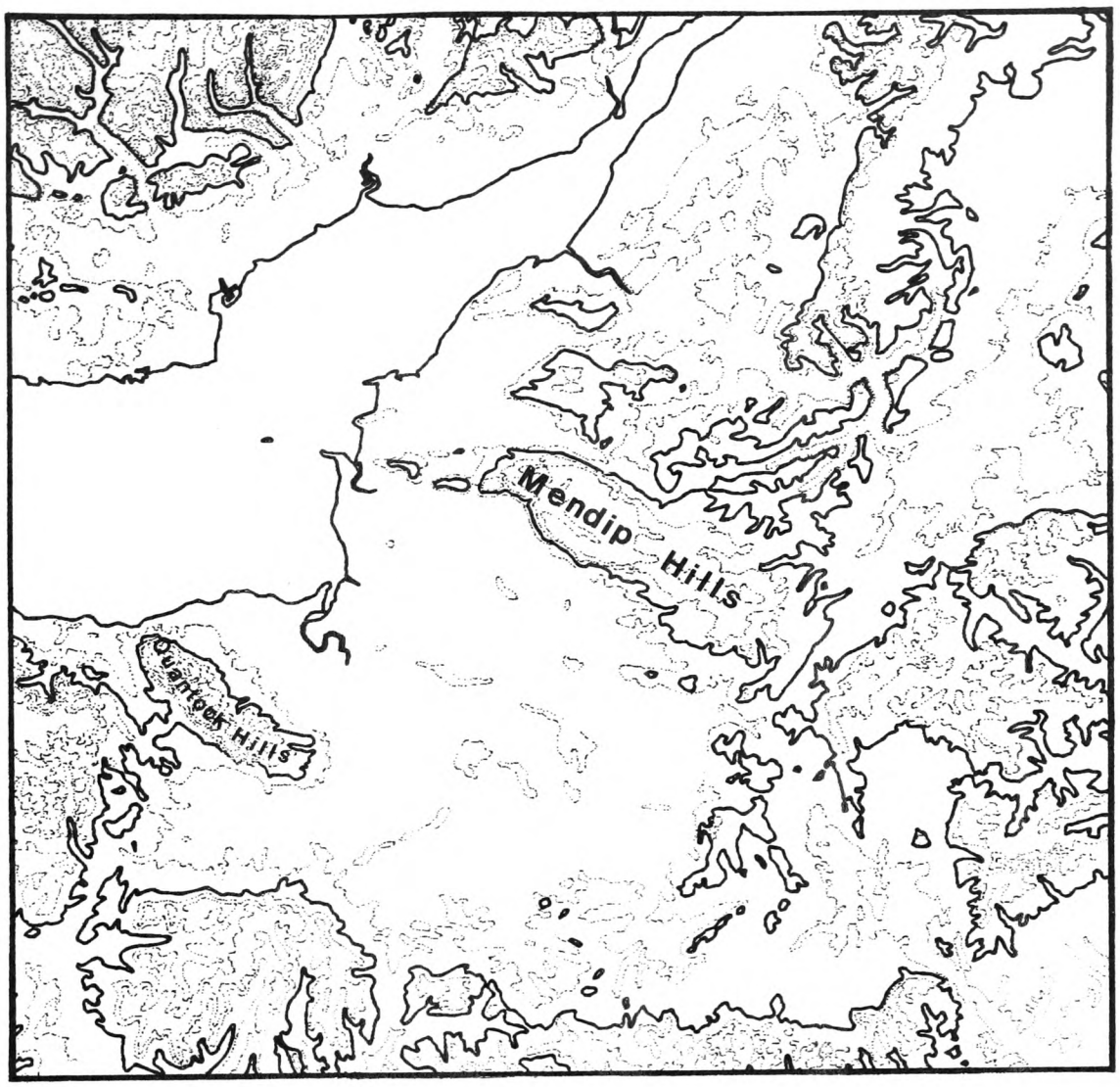


Fig 3.1 Using Ordnance Survey Route Planner map contour data (Library grid square 201) to identify highland area regions with a threshold (thick line) set at 200 feet.

3.3 RASTER DATA

3.3.1 INTRODUCTION

A major advantage of raster data over vector data is that it enables any arbitrary region on the map to be examined without having to read through several vector data records. This is very useful for name placement, where there is a need to put a name into empty space. Two methods of storing raster data will be considered, these are grid array and run-length encoded data.

3.3.2 GRID ARRAY

In the case of raster data held in grid array form, each of its elements or pixels, represents the contents of the equivalent location on a map. If a feature is present on the map at that location, then the pixel value is set to a non-zero number which indicates the class of feature present. Yoeli (1972) and Basoglu (1984) use such an array for detecting overlaps between labels and underlying features or placed labels.

The size of the raster image must be large enough, in terms of pixels, so that fine enough resolution is possible for name placement techniques to work properly. This implies that the pixel size must be that of the

smallest letter and preferably smaller, to allow adequate representation of line thickness in the raster image. In the case of the Route Planner map name placement system, LABPOS (Section 6.3), raster data covering a 100x100km square region, is typically rasterized into a grid size of 512x512. This has a ground pixel size of 195m or 0.3mm on a 1:625000 scale map. The smallest letter size is 1mm and the smallest line gauge is approximately 0.2mm.

A typical two dimensional integer array, on for example a VAX computer, contains NxN 32 bit integer numbers. However, taking into account that, on average 90% of most maps are empty space, and each pixel needs only one bit in order to indicate the presence of a feature on the map, then most of the available memory in the integer array is not utilised.

A more sophisticated approach to storing just one bit per pixel is to allow for storage of different feature classes in parallel bit planes in the raster image array. For instance, in name placement, one needs to know what classes of feature a name will obscure if it is placed at a certain location on the map. Clearly, representing map features as just one bit will not be of much use in identifying classes of feature, therefore one can make use of a method of overlaying bit planes. This technique can be used to store up to 32 separate cartographic classes, held in parallel inside the same integer array. Freeman

(1985) refers to the use of "overlay planes" for storing different kinds of information, which can then be assembled for a particular map. Cook (1986) makes use of bit planes (Fig 3.2) for storing classes of feature according to levels of importance to the name placement process. Each bit plane has an associated priority value indicating importance, the higher the number the more important. Each level can be accessed by the following logic formula:

$$\text{pixel}(X,Y,N) = \frac{(\text{pixel}(X,Y) \wedge 2^N)}{2^N}$$

Where **N** is the bit plane number (0-31).

X is the sample coordinate of pixel.

Y is the line coordinate of pixel.

pixel(X,Y) is pixel integer value at location X,Y in the raster image

pixel(X,Y,N) is the bit value of a pixel at location X,Y in the raster image and in bit plane N.

∧ is logical AND.

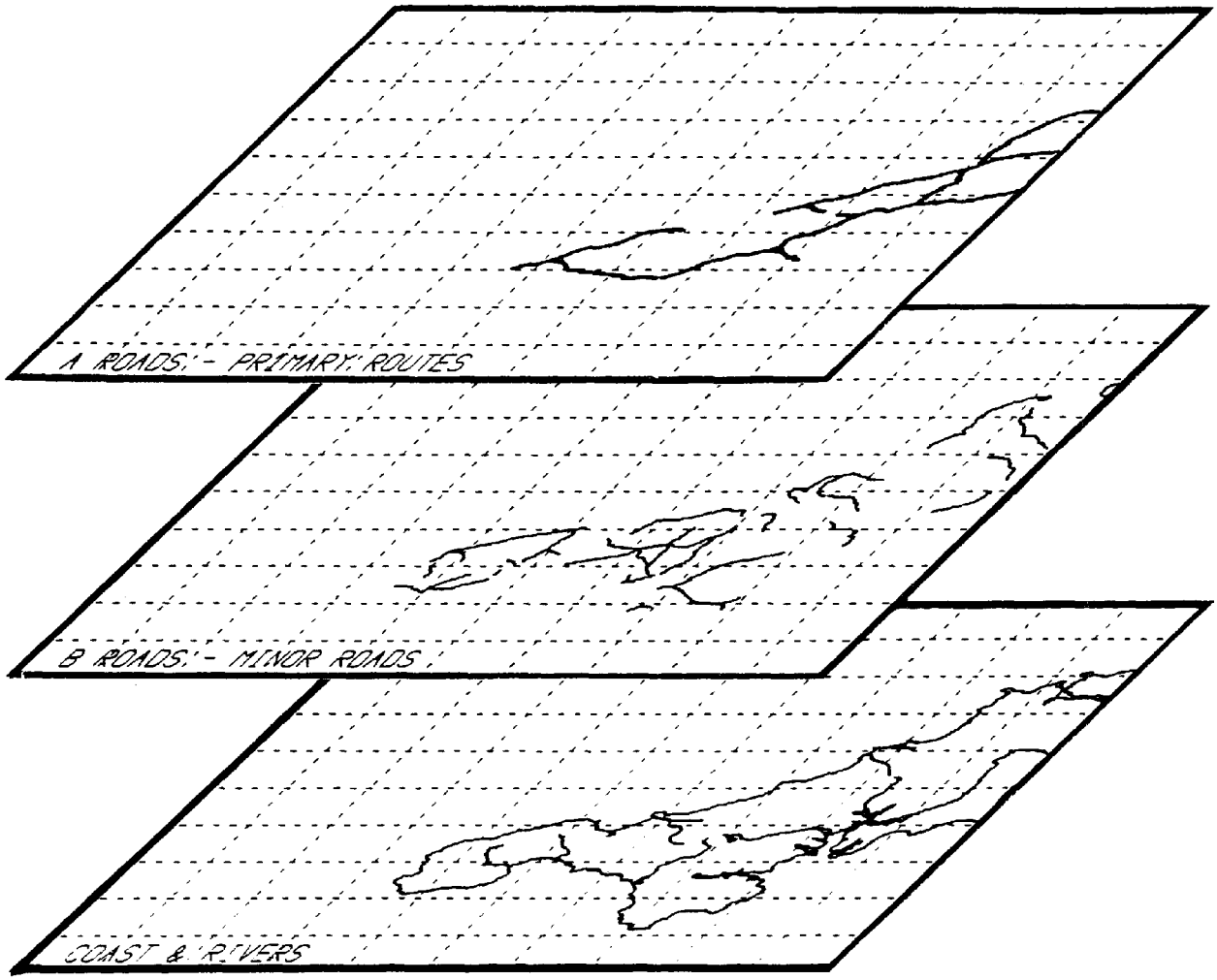


Fig 3.2 The use of raster bit planes for storing feature classes.

3.3.3 RUN-LENGTH ENCODING

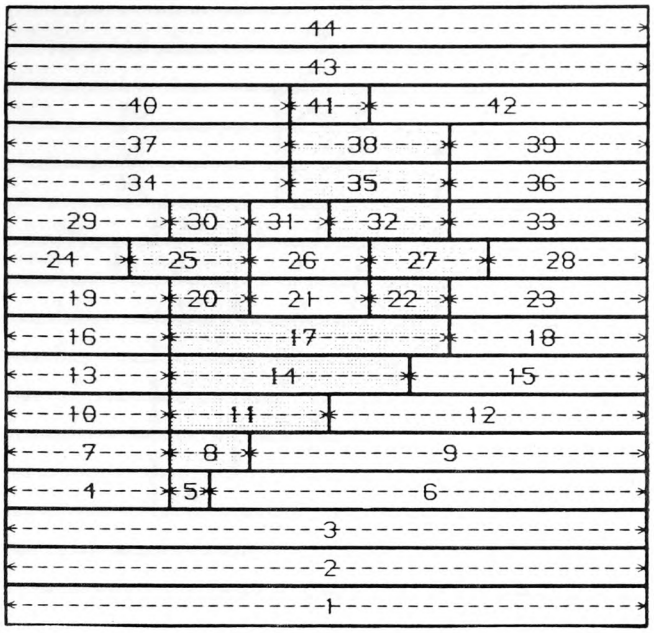
Even on a bit plane rasterized image, the vast majority of pixels are empty due to the need to represent large blank regions on the map in between features or large uniform areal features such as seas and lakes. Such vast regions of adjacent identically valued pixels constitute a waste of available memory and simple data compression techniques can be used to reduce memory storage.

One such data compression technique is called run-length encoding, and can take several forms. The simplest form, called row order (Mark and Goodchild), records the value of the pixel in the bottom left hand corner of the image, and then records the number of pixels one can count horizontally before the pixel value changes or until the end of the line is reached. The process is repeated for the next pixel and neighbouring pixels along the run-length or begins on a new line if necessary (Fig 3.3). A version of this run-length encoding, which has been applied to Ordnance Survey 1:625000 database by the author, has been shown to permit a memory space saving of typically 70% in high feature density map areas when compared to grid array raster data. In very low feature density regions of the map, space savings of up to 98% have been observed.

Another slightly more effective form of run-length encoding uses the row-prime order (Mark and Goodchild) which scans to the end of a line and without breaking the strip, moves onto the next line and reverses direction (Fig 3.4). The purpose of this is to minimise the changes of pixels values encountered previously when moving to the start of each consecutive line.

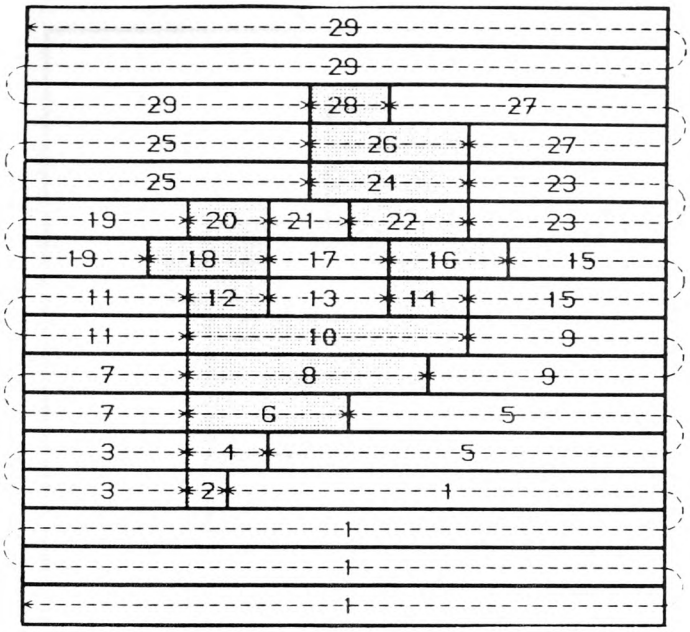
There is also a discontinuous form of run-length encoding, whereby the most common pixel value such as the sea or blank map space, is ignored, and run-length encoding only applies at the start of pixel values other than the background (Fig 3.5) (Jones, 1987). Needless to say the line and sample locations of the start of each run-length have to be recorded at extra storage expense.

Finally, an even more effective data compression can be achieved by using a scanning technique which stays within the areas of expansive regions of similar pixel values, for as much of the run-length as possible. The "Hilbert-Peano" order (Fig 3.6) has this property and traces out a recursive curve which has a high space filling property (Mark and Goodchild). It can cover much smaller regions of the map for a given length than any of the other run-length encoding methods discussed.



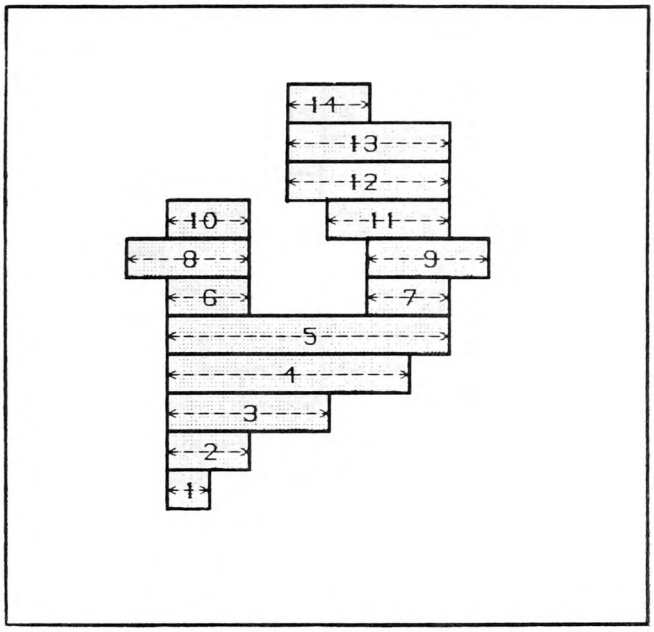
Strip Number	Strip Length	Strip Value	Strip Number	Strip Length	Strip Value
1	16	0	23	5	0
2	16	0	24	3	0
3	16	0	25	3	1
4	4	0	26	3	0
5	1	1	27	3	1
6	11	0	28	4	0
7	4	0	29	4	0
8	2	1	30	2	1
9	10	0	31	2	0
10	4	0	32	3	1
11	4	1	33	5	0
12	8	0	34	7	0
13	4	0	35	4	1
14	6	1	36	5	0
15	6	1	37	7	0
16	4	0	38	4	1
17	7	1	39	5	0
18	5	0	40	7	0
19	4	0	41	2	1
20	2	1	42	7	0
21	3	0	43	16	0
22	2	1	44	16	0

Fig 3.3 Row order run-length encoding.



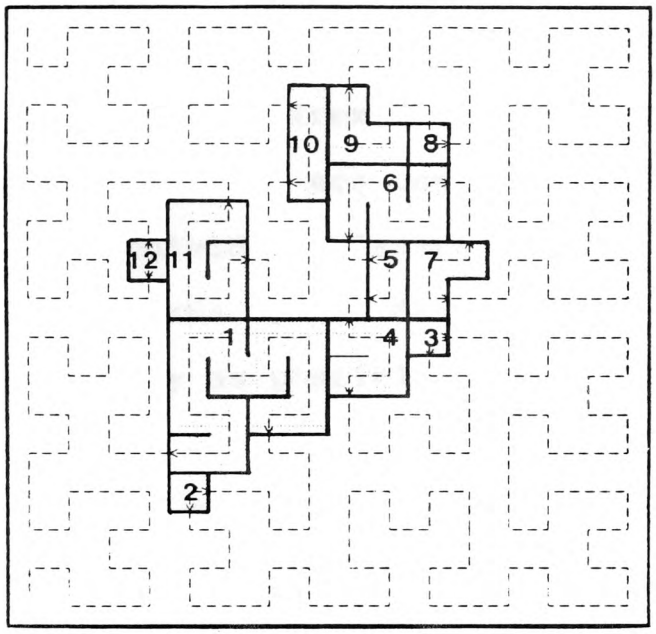
Strip Number	Strip Length	Strip Value	Strip Number	Strip Length	Strip Value
1	59	0	16	3	1
2	1	1	17	3	0
3	8	0	18	3	1
4	2	0	19	7	0
5	18	0	20	2	1
6	4	1	21	2	0
7	8	0	22	3	1
8	6	1	23	10	0
9	11	0	24	4	1
10	7	1	25	14	0
11	8	0	26	4	1
12	2	1	27	12	0
13	3	0	28	2	1
14	2	1	29	39	0
15	9	0			

Fig 3.4 Row-prime order run-length encoding.



Strip X pos.	Strip Y pos.	Strip Length
5	4	1
5	5	2
5	6	4
5	7	6
5	8	7
5	9	2
10	9	2
4	10	3
10	10	3
5	11	2
9	11	3
8	12	4
8	13	4
8	14	2

Fig 3.5 Discontinuous run-length encoding.



Strip X pos.	Strip Y pos.	Strip Length
5	5	14
5	4	1
11	8	1
9	7	4
10	9	2
9	11	6
12	10	3
11	13	1
10	13	3
8	14	3
6	10	6
4	10	1

Fig 3.6 Hilbert-Peano (discontinuous) run-length encoding.

3.3.4 STORAGE OF RASTER DATA FOR NAME PLACEMENT

One overwhelming requirement, for name placement purposes, is that bit planes are used in order to allow the separation of individual classes of features. Another specification is that the raster data can be encoded and reconstituted as easily as possible.

The Hilbert-Peano order run-length encoding technique requires a raster size of side length two to the power of N (where N is a positive integer number). This is a disadvantage for a name placement system where careful tuning of the raster size may be needed to give a careful balance between fine enough raster resolution and the constraints on memory.

On comparing the raster data formats, run-length encoding has significant space savings advantages over the grid array format. Although some of the more sophisticated forms of run-length encoding are very compact, it was decided to select "row order" run-length encoding since it is the simplest form of compressed raster data to load (Section 7.5.2).

3.4 CHAPTER DISCUSSION

An automated name placement system capable of placing names on a wide range of maps should ideally contain both vector and raster data if it is to cope with the wide range of rules discussed in chapter two. The vector data is suitable for accessing individual features and provides a reference system for the placement of a label relative to its feature. It also provides key topological information such as the link structure making up lines, which can be used to decide where to repeat labels along a line (Rule [2.32]). Raster data is useful for looking at any potential label position on the map quickly to analyse underlying map features before a name is placed.

Point data should be stored in the form of point locations, and line data represented as vectors. It was decided to store individual area information as raster data in a run-length encoded form which is both easy to access, and compact. Additionally, this is a suitable format for placement of area labels, where raster data will be needed anyway (Section 5.4 and 7.4.4). Undefined areas were discussed in section 3.2, and although in the case of "hills" contour data may help to define the extent of such regions, the approximate extent of the area has to be stored in the database if such features are to be labelled.

All three data types serve the purpose of providing the source data necessary for the production of the raster map image. They are also used in the name placement process for identifying the unique positions of labels against the raster image, during feature masking (Sections 6.5.3 and 7.5.5).

CHAPTER 4

REVIEW OF AUTOMATED NAME PLACEMENT METHODS

4.1 INTRODUCTION

The purpose of this chapter is to give a general review of automated name placement systems and describe which techniques from these are applicable or adaptable in the context of the author's research. It will also highlight the restrictions of these name placement systems and outline the problems that remain to be solved.

Just over a dozen automated name placement systems have been implemented and published since Yoeli (1972) investigated the basic concepts of automatically placing names on maps. Of these, several papers were either published or were discovered mid course in the author's three years of research and so have not had much bearing on the research. The ideas contained in some papers will therefore only be briefly discussed in this chapter (Van Roessel, 1987), (Zoraster, 1986), (Zoraster and Bayer, 1987).

Most fully integrated name placement systems involve three discrete stages:

i. Selection of labels and their configurations.

ii. Iterative placement of names to find an overall solution.

iii. The actual placement.

Label selection involves choosing from a comprehensive list of named features those suitable for labelling in order to satisfy map name distribution and name density criteria. Configuration selection forms part of label selection in that the assignment of label sizes, styles, and the placement arrangement of the name with respect to its feature are normally decided before placement. The selection of labels and configurations are defined according to a set of rules related to the class of the features and the map type. Humans and most automated name placement systems alike place names iteratively to find the optimum placement satisfying general and specific name placement rules such as those discussed in Chapter two. Once an optimum solution has been found the system must output the label configurations, sizes and positions.

Yoeli developed the first automated name placement system but because of the limited hardware available, his system only tackles horizontal point and area names, leaving the remaining few curved names and names which could not be placed due to conflict, to manual placement.

Relatively few researchers have tackled the problem of name selection. Basoglu's NAMEPL (1984) applies a sophisticated name selection process prior to placement. He continues to develop Yoeli's ideas with the inclusion of line and area placement but also introduces the concept of placing labels in order of importance. Another name selection system is Langran and Poiker's PLACENAMES (1986) which, although only applicable to point names, does include an ability to delete labels during placement if necessary.

Although most name placement systems place names iteratively by testing label positions systematically, some have the ability to backtrack during name placement: Greggains (1982), Freeman and Ahn (1984), Mower (1986) and Jones (1987). Hirsch's system (1982), although limited to point labels, is the first and only system where vector accumulation for conflicting labels is used to show which way to move to avoid overlap.

Expert systems have also been applied to name placement. For instance Freeman and Ahn's (1984) rule-based system, AUTONAP, places names in order of size or importance and can request the deletion of point features or a change in font size if a placement attempt fails. Pfefferkorn et al (1985) describe a research prototype expert system, ACES, which uses an object-orientated knowledge representation scheme. The system selects symbol placement, font type, label size, and level of description which best fits the map class.

Four aspects of automated name placement systems will be considered in this chapter:

i. Selection of labels and their configurations.

ii. Algorithms for generating label positions.

iii. Label overlap detection.

iv. Strategies for solving the name placement problem.

4.2 SELECTION OF LABELS AND THEIR CONFIGURATIONS

4.2.1 INTRODUCTION

Many authors of papers on name placement systems ignore name selection and assume that all names supplied can be placed. However this is not always the case and so a decision must be made as to which names to select prior to placement. The alternative involves names being deleted during placement if the name density is too high for a given area of the map. Because of the computational overheads, the former selection process is preferred although a combination of the two may be necessary.

A second form of selection applies to the choosing of the configuration for each label. Configuration in this respect refers to not just placement arrangements, but also label size, style etc.

4.2.2 LABEL SELECTION

Langran and Poiker (1986) state that human cartographers arrange names on maps as they are selected and that selection criteria relate to feature importance. The maximum number of letters which can be placed on a map is approximately given by the ratio of free map area to the area of the basic letter size (Yoeli, 1972). In

practice this number is considerably less because names must not cross features of the same colour and there is a need to avoid ambiguity [Rule 2.3].

In the name placement system design proposed by Hirsch and Glick (1983), the selection of both features and names is achieved by the user interactively specifying selection parameters such as scale and map theme etc. Names and spatial data are then automatically retrieved with the possibility for the user to manually edit names further.

Basoglu (1984) has developed a sophisticated name selection system for atlas type maps. In order to decide which names to select at a particular scale he specifies criteria for selecting names from a database. The point component of the database includes information on population, remoteness, number of airports, number of first class roads, number of historical symbols and the point coordinates. This information can then be used to assign a rank to each point. Basoglu sorts his point names into an order of rank, computes how many names should be present on a map at the specified scale using the Topfer-Pillewitzer selection criteria and then attempts to place this number of names starting with those of the highest ranks. Both the line and area components of the database include the pointer to the line coordinate data, the type of feature (lake, river, island or area) and the

scales below which the features and labels are not plotted.

Langran and Poiker (1986) regard label selection as a form of generalisation because settlements are typically the most commonly named features and selecting such names is similar to selecting settlements. In their name placement system, "PLACENAMES", an algorithm called Quadrat Reduction is used to select which settlements to label. This involves dividing the map into a matrix of cells and defining the number of settlements selected per cell according to its global density parameter. This can be adjusted to suit the specified scale and purpose of the map. The matrix cells are then filled with all the available points so as to provide information on point density per cell. The matrix is then scanned from the cell with biggest settlement downwards in size. If the number of points in a cell exceeds the global density of points for the map then, cells immediately adjacent to the overcrowded cell are checked for available space into which their influence may overflow. Surrounding cells with less than the global density are credited with fractions of the overflow settlements. A point density of one allows a cell to contain five settlements if all its adjacent cells are vacant. A half is credited to each of the 8 adjacent cells and one to the original cell.

If the original cell still exceeds the maximum density after its adjacent cells have been checked and credited, settlements have to be deleted. Settlements in the centre and adjacent cells are all potentially deletable. Settlements are removed in order of increasing rank until the global density parameter is reached.

Langran and Poiker's (1986) system has the advantage that it processes metropolitan areas as units and preserves regional clustering. Since urban areas are processed first, most empty cells are credited with their overflow by the time rural cells are processed, thus enforcing lower rural settlement densities.

4.2.3 LABEL CONFIGURATION SELECTION

All authors select label configuration prior to placement. Two examples are given here of the selection of label sizes:

1) Langran and Poiker classify settlements according to population size and assign an appropriate letter size for each class.

2) Basoglu specifies a minimum letter or type size of approximately 4 points. He also finds that type size can be linked to map scale:

type size = (input scale/output scale).constant type size

4.2.4 SECTION DISCUSSION

All the name selection techniques discussed involve the selection of labels and configurations prior to placement. This sequence of events will be used in the author's research. However, since the main purpose of the research is to investigate name placement rather than name selection, only a limited form of name selection will be attempted (Sections 6.4, 8.3.3, 8.4.3 and 8.5.3). Nevertheless an ideal name placement system capable of placing names on a large variety of maps should have the capability of performing label selection according to criteria such as those discussed in this section of the chapter.

4.3 ALGORITHMS FOR GENERATING LABEL POSITIONS

4.3.1 INTRODUCTION

The human cartographer has several advantages over an automated name placement system. For instance he has a visual image of the map and can quickly locate an open area, anticipate its use and place names accordingly. In an automated system, the placement of labels occurs in a serial fashion and to reduce search time placement positions are usually quantised into a small number. A variety of placement algorithms have been developed for each of the three name types. The algorithms make use of either vector or raster data to achieve placement.

4.3.2 POINT NAMES

4.3.2.1 INTRODUCTION

Two techniques have been utilised for point name placement. The first uses a raster map where the pixels form a grid cell structure and names are placed in grid cells around the point. The second allows the name to be placed on a circle, centred on the point, with a user specified radius of proximity. Although most authors cater for simple single lined labels, Freeman and Ahn (1984), Pfefferkorn (1985) and Cook (1986) allow for the splitting

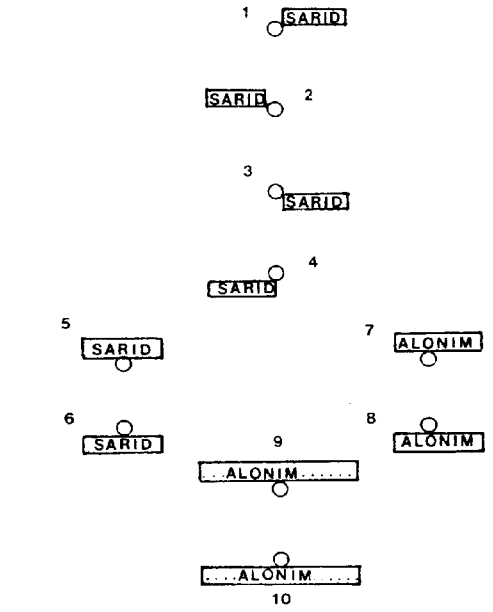
of names into separate lines.

4.3.2.2 GRID CELL TECHNIQUE

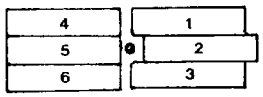
Yoeli divides the map into a grid of cells, each cell corresponding to a character size. The arrangement of point names around a point is shown in Fig 4.1a. Basoglu, uses the same arrangement of point names but with a much finer grid of cells. Jones (1987) stores labels as pairs of run-length strips in PROLOG and the arrangement of names can be seen in Fig 4.1b.

4.3.2.3 RADIUS OF PROXIMITY TECHNIQUE

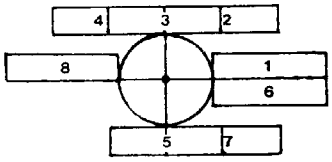
Hirsch (1982) introduces the concept of placing the label at a fixed distance from the point symbol around an imaginary circle which he calls "constrained displacement" and the fixed distance is equal to a letter height (Fig 4.1c). A set of eight preferred positions around the circle is used and a name may also be placed at any intermediate position. All positions are expressed as angles.



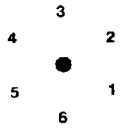
(a) Yoeli, 1972.



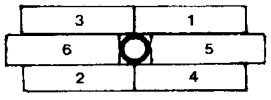
(b) Jones, 1987.



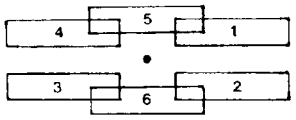
(c) Hirsch, 1982.



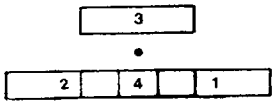
(d) Greggains, 1982.



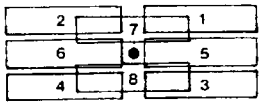
(e) Freeman and Ahn, 1984.



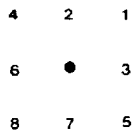
(f) Cromley, 1986.



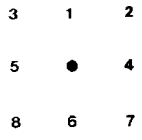
(g) Langran and Poiker, 1986.



(h) Mower, 1986.



(i) Zoraster, 1986.



(j) Zoraster and Bayer, 1987.

Fig 4.1 Point name placement preferences.

Although the number and arrangement of positions used by different authors can vary considerably (Fig 4.1a-j), it is of interest to note that Langran and Poiker (1986) use only four label positions which they claim are the likeliest positions to solve most label conflicts. The positions are widely separated in order to ensure a radical change in location each time a name needs to be moved (Fig 4.1g). Only using four positions has the advantage that, because the choice of positions is smaller, a name placement solution can be found faster.

In Zoraster's (1986) system two different types of point labels are used, those belonging to wells which have eight positions (Fig 4.1i) and shot point labels which lie at intervals along seismic shot point lines. The latter are placed perpendicularly to the lines and only have two positions corresponding to each side of the line.

4.3.3 LINE NAMES

Relatively few authors have tackled the problem of line placement. This is probably due to the fact that line labels can be of any length and can have any degree of wiggliness making the placement of labels very difficult to parameterize. Nevertheless, four methods will be briefly outlined.

Basoglu's method for computing the position of a line label (Fig 4.2) firstly involves filtering the line to remove unnecessary wiggliness. It then finds two consecutive filtered coordinate pairs with the widest separation and fits a second degree curve through the unfiltered coordinate pairs lying between these two points. Next the largest deviation from the base line, defined as the line between the two filtered coordinate pairs, to the unfiltered coordinate pairs is found and the name is offset by this amount and placed parallel to the curve.

In AUTOTEXT (Greggains, 1982), river labels are placed at six positions along the line spaced at alternate half label intervals centred on the mid-point of the line and aligned with the river.

Freeman and Ahn (1984) describe how AUTONAP performs line placement by dividing each line up into segments of fixed length. Starting at a certain distance from the end of each segment, a linear search is made over the extent of the segment until all possible positions have been tested for overlap. In determining whether a line position is good, Freeman and Ahn take into account the label's distance from the centre of the line segment and secondly the curvature.

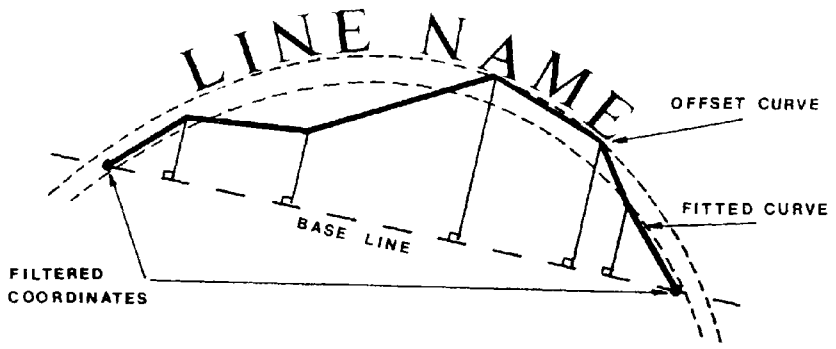
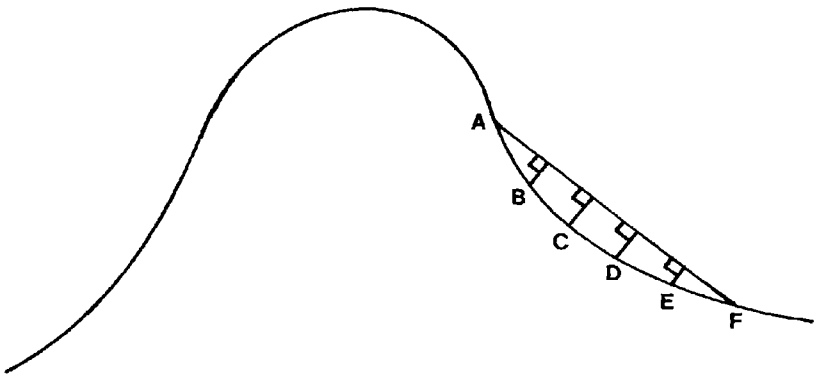
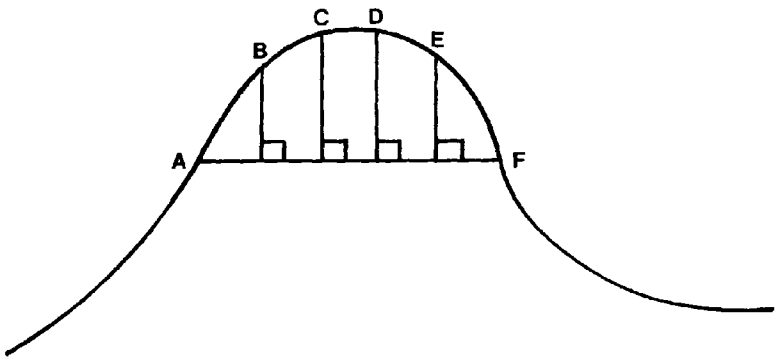


Fig 4.2 Basoglu's (1984) line name placement.



"Straight Enough"



"Not Straight Enough"

Fig 4.3 Sampson's definition of whether a section of a line feature is straight enough to place a label on. This occurs when all the points between A and F have a perpendicular distance of less than half the labels width (Davis and McCullagh, 1975).

Sampson (Davis and McCullagh, 1975) explains how to determine whether a section of a contour line is "straight enough" to place a label on. This occurs in Fig 4.3 when AF is equal to the label length and all the points between A and F have a perpendicular distance of less than half the label's width.

4.3.4 AREA NAMES

Area names are considerably more difficult to parameterize than line labels because it is permissible to vary the label size and shape to conform to that of the area (Rule [2.42]). Most authors therefore severely restrict the choice of label positions or configurations.

Yoeli's system places horizontal area labels centred inside a bounding rectangle surrounding the area (Fig 4.4). If the letters can fit inside, they are spread out across the rectangle and cross the centre of gravity. Because this can occasionally result in the label falling slightly outside the area, the bounding area rectangle is made smaller to avoid this. Internal area labels are placed so that the distance of extreme letters from the border is twice the letter separation distance.

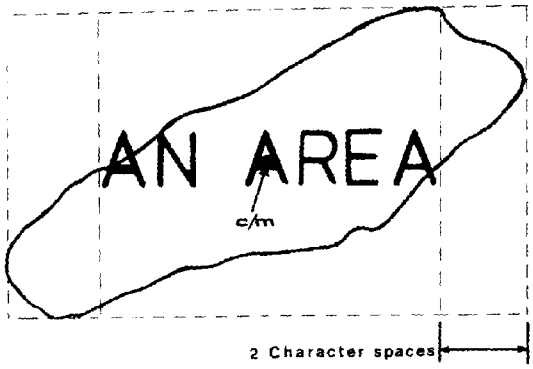


Fig 4.4 Yoeli's area name placement.

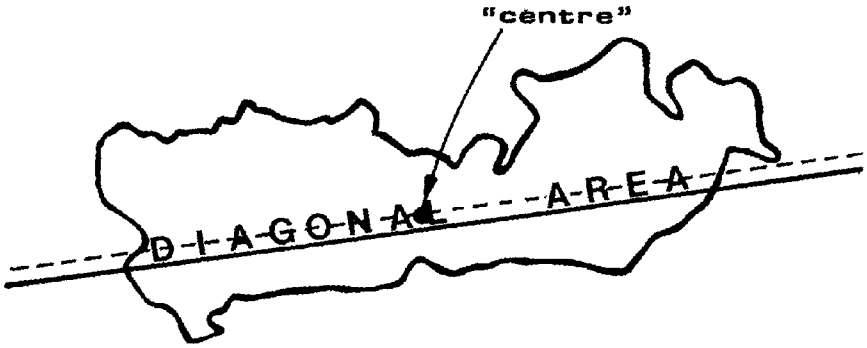


Fig 4.5a Basoglu's (1984) area name placement - "Land".

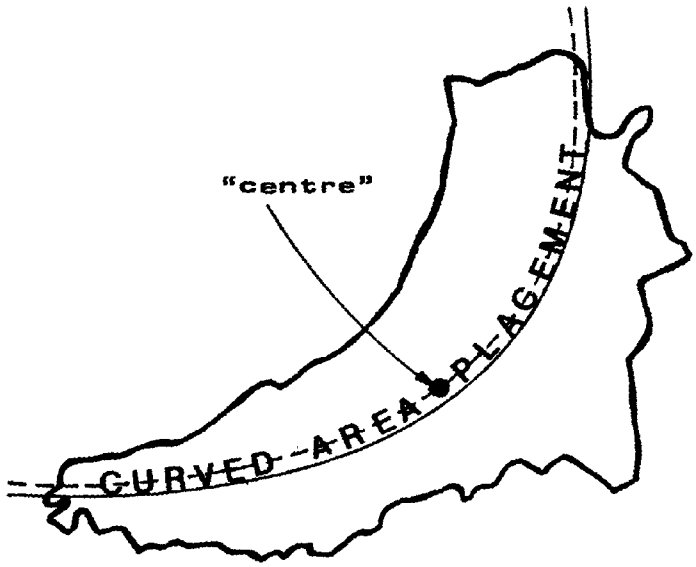


Fig 4.5b Basoglu's (1984) area name placement - "Other area".

Basoglu's system for computing the position of an area name (Fig 4.5a & b) firstly filters the area boundary coordinates to remove unnecessary wiggliness but maintains the area's major characteristics. If the area concerned is a land area then a line is fitted through the filtered points and the name is placed parallel to the line but offset so that it passes through the "centre" of the area. If the area is not of the land type then a second degree curve is fitted through the filtered points and the name is placed parallel but offset so that it passes through the "centre" of the area. The letters of the name are placed at equally spaced intervals inside the area allowing for a gap at each end to prevent the letters crossing over the area boundary.

In Greggains' system, the minimum and maximum limits of an area feature are found and from these the approximate centre of the feature (Fig 4.6). The first label position is centred on this, the remaining positions are then placed on a circle around this location with a radius of two characters, starting at 0 degrees and moving anti-clockwise with 72 degree increments.

Hirsch and Glick (1983) find the minimum area enclosing rectangle for the area concerned and divide this up into equispaced line segments corresponding to the number of characters in the area name (Fig 4.7). The mid-point between the area boundary in each line segment

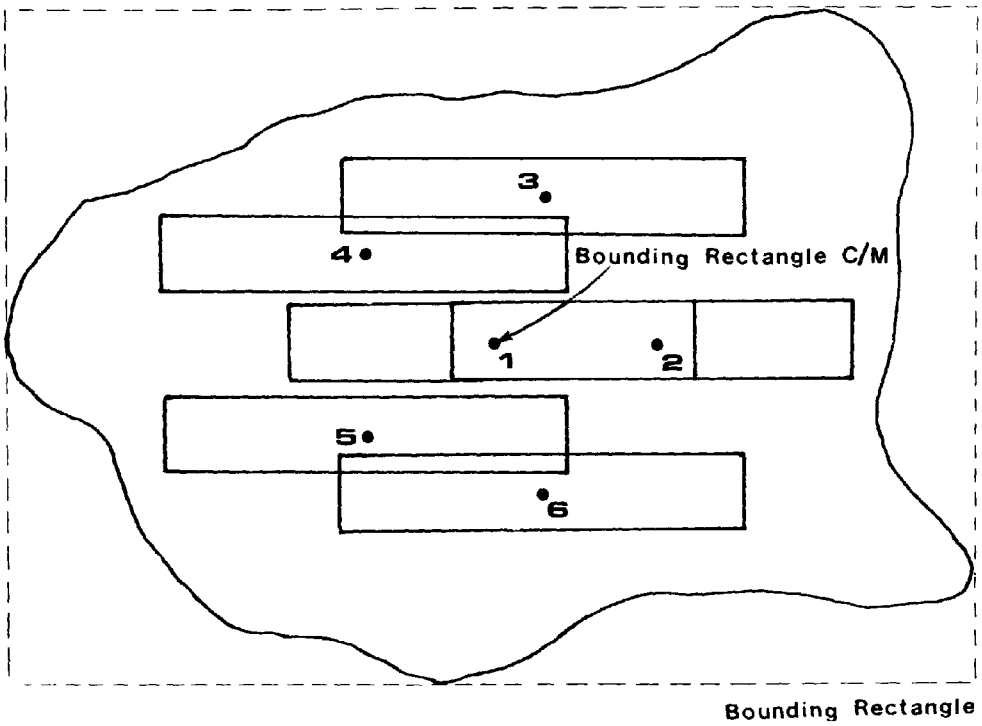


Fig 4.6 The author's interpretation of Greggains' area name placement (1982): labels placed at 72° intervals around the centre of the area horizontal bounding rectangle and offset from this by 2 character spacings (except for the first position).

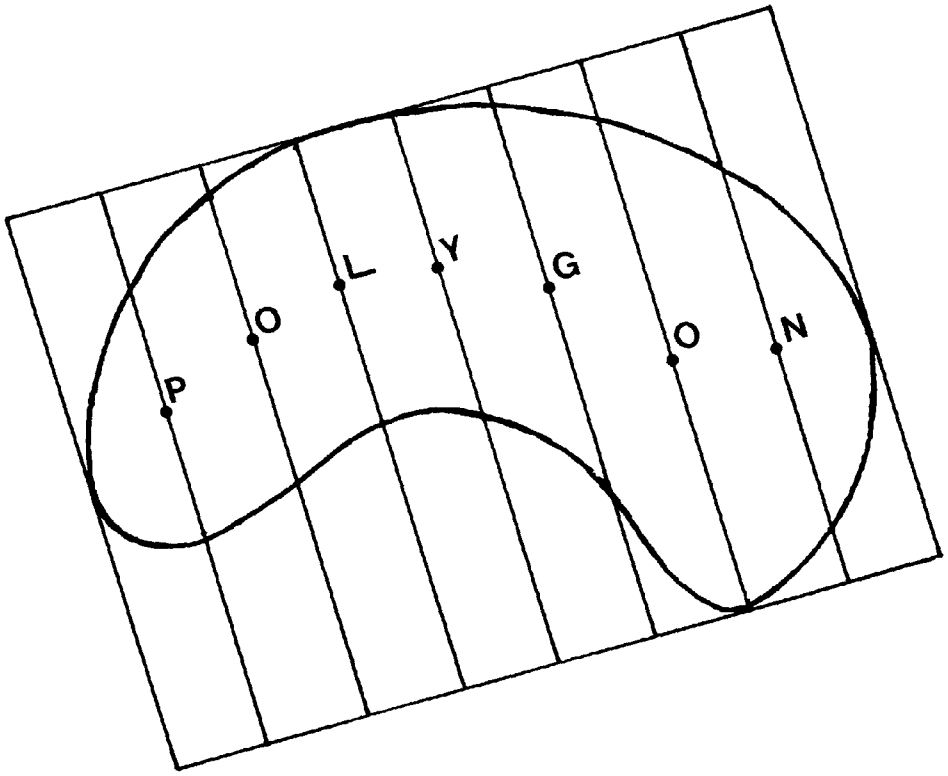


Fig 4.7 Hirsch and Glick's area name placement (1983).

is found, this determines the position of each character in the area name. The orientation of each character can either be parallel to the X axis or to a line fitted through the mid points.

Freeman and Ahn (1984) have built the most sophisticated area placement algorithm. They first apply a polygonal approximation to the area boundary to remove small bays and peninsulars. A skeleton is then generated and a threshold is applied to remove sections of the skeleton less than the character size. All sections except those associated with the greatest area are removed. Of the remaining skeleton a placement zone is applied around each non-disjoint subsection. These then form the areas in which the name can be placed. If adjustments are required to an area name, this is achieved by shifting the name perpendicularly to the name direction or by altering the spacing between the letters. Alternatively, if the name is too large, it can be fitted outside.

Van Roessel (1987) points out that if the centroid of an area lies outside, then it is no good using a method which relies upon the placement of a name on the centroid. He suggests a different approach whereby if one considers the label as a rectangle, it must be wholly contained inside the area at possibly several positions (Fig 4.8). To achieve this he divides the area into horizontal strips with the edges at each of the area boundary vertices. Once

a set of strips has been generated, these are then used to find the maximum sized rectangle that can be fitted inside the area.

4.3.5 DISCUSSION OF POSITION GENERATION ALGORITHMS

Algorithms which quantise name placement positions into a finite number are particularly useful for automated name placement algorithms since these allow label positions to be examined in a systematic way. This is achieved successfully for point labels where most authors make use of the radius of proximity method of placement. Line placement is complicated by the fact that placement of labels on curved sections is regarded as poor positioning. Nevertheless, Freeman and Ahn, and Basoglu appear to have successfully tackled the problem of curved line placement. Greggains suggests a method for placing labels at regular intervals along a line which is worthy of consideration due to its simplicity.

Area labels with all the associated degrees of freedom of placement have a variety of algorithms available. Basoglu offers an algorithm capable of generating one curve on which to place the label. Freeman and Ahn have successfully demonstrated their use of sections of area skeletons on which to place the label. However only Greggains suggests a method whereby area

label placements can be quantised into a finite number in a similar way to point and line placement.

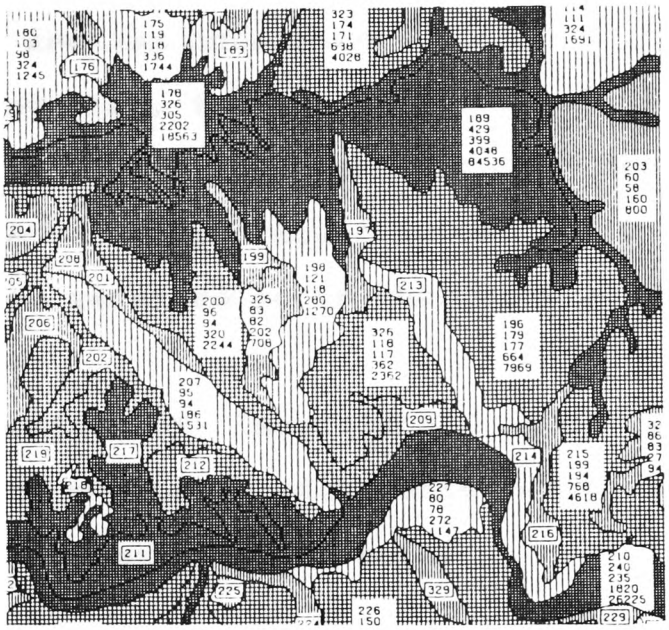


Fig 4.8 van Roessel's area name placement (1987).

4.4 LABEL OVERLAP DETECTION

4.4.1 INTRODUCTION

One of the essential aspects of name placement is that labels should not "overlap" each other (Rule [2.1]) and should avoid overlapping underlying features (Rule [2.3]). The former overlap condition is very rarely broken (It is in fact broken in Fig 2.3), the latter condition is more flexible. Another constraint, although not strictly to do with overlap, is that labels of similar size must not be placed too closely together or too near to other features since this can result in ambiguity (Rule [2.3]) and this will be referred to in chapter 6 as "conflict".

The detection of label:label and label:feature overlap can be performed with raster data or vector data and sometimes a combination of both.

4.4.2 DETECTION OF OVERLAPS USING RASTER DATA

Yoeli's (1972) system was the first to make use of raster data or grid cells. This serves the purpose of indicating regions of the map where labels cannot be placed for example on top of point features. Every time his system places a label, a "free" zone is added around the edge of it in the raster data to avoid other labels

being placed too close. Langran and Poiker (1986) also make an allowance for such a buffer zone around a label and Cromley applies this around each point. Unfortunately, because Yoeli's system restricts the placement of letters to grid cell squares which are equal in size to each character size, this severely restricts the placement freedom of labels and is likely to lead to occasions when labels are being flagged as unplaceable when a slight shift in position of less than a grid cell square would find a solution. Basoglu uses a much finer version of Yoeli's grid cells which he calls a "bit matrix" and allows for the inclusion of all three features types. However the idea of using grid cells to ban the placement of labels over any feature is flawed because it is often permissible to place labels over features of low importance if no other positions can be found.

In Greggains's (1982) system, a pixel map, similar to the bit matrix, is used with the effect that different objects can be made distinguishable by their pixel values. The higher the pixel value, the more undesirable it is to place labels over such features. Greggains defines a threshold value making it illegal to place labels over pixels containing such values. This approach is also adopted by Jones (1987) except that the background map contents are stored in run-length encoded lists and the labels and point features are stored as separate individually accessible run-length encoded masks. Jones

(1987) uses a pixel size slightly greater than half a letter height so as to ensure a buffer zone between each label.

Freeman and Ahn's (1984) AUTONAP divides the map into grid cells and uses pointers to identify points and labels which fall inside each cell. A "free space" list is constructed for each label so as to indicate which cells can be used for placement without overlapping placed labels or point features. The grid cells form part of a 100x100 cell array across the map.

4.4.3 DETECTION OF OVERLAPS WITHOUT RASTER DATA

In Hirsch's (1982) system, overlap detection is performed without the use of grid cells and instead relies upon sorting "map objects", consisting of labels, points and the map boundary, by their northing coordinate.

To check if a label is in overlap, his system searches the objects above the label position and continues until either an overlap is detected or the top of the current label is below the next highest object and hence no further checks are necessary. The method of detecting overlap involves testing the four corners of each rectangular label to see if any of them are contained within a radius of proximity circle or a label rectangle

or a boundary area.

The disadvantage of Hirsch's technique is that there may be many objects within the same northing range as the label, which will need to be tested but which are too far east or west to have any chance of overlap.

In Cromley's system (1986), label conflicts are found by numerical searches, using a Thiessen diagram for point distribution generated prior to name placement. The search for overlaps is conducted sequentially by making comparisons between a given point feature and its Thiessen neighbours.

Zoraster and Bayer (1987) use a single sweep algorithm for detecting overlaps between line segments constituting a label bounding rectangle or another object. All objects are sorted according to their easting coordinate and then subjected to a hierarchy of tests for overlaps of other items within this range.

Langran and Poiker use a pointer array for storing label overlaps but it requires re-sorting each time a label is moved. It points to labels in order of increasing northing coordinates.

4.4.4 HIGH LEVEL OVERLAP INFORMATION

Greggains uses a combination of a pixel map matrix and feature bounding rectangles for detection of overlap. The bounding box enables a quick check for overlap and the pixel map a definitive check. A conflict table with a record structure consisting of conflicting label pairs is built by testing each label pair in all combinations of positions. The corners of each character are calculated and the underlying pixels on the map are checked for conflict and an undesirability index for that position calculated. Because Greggains only considers six positions for each label, at most a conflict between two labels would have 36 entries in the table. However if the system were expanded to include between ten and twenty label positions then the number of entries in the conflict table would become excessive.

In Freeman and Ahn's (1984) AUTONAP, a graph is used to represent overlapping labels where a node in the graph represents a point name. If two labels overlap, their nodes are joined. To avoid comparing every node against every other node, nodes are sorted in order of increasing Y values, then only nodes which fall within a fixed Y range need to be examined.

In the approach adopted by the author of this thesis (Cook, 1986), before name placement takes place, a test is

made to see which labels have the potential to overlap with each other. This knowledge cuts down on the search time needed for detecting overlaps later on during name placement.

4.5 NAME PLACEMENT STRATEGIES

4.5.1 INTRODUCTION

Most of the strategies that are about to be described only apply to point features, however those which apply to all three types of label place these in order of areas, points and lines. Name placement strategies fall into four categories:

1) Non-iterative

2) Iterative

3) Backtracking

4) Expert Systems

4.5.2 NON-ITERATIVE

This forms the crudest form of automated name placement system in that once a label has been placed, it cannot be moved. Initially when attempting to place a label, it is tested in sequence of priority ordered positions until either it finds one where there is no overlap or until no more positions remain to be tested, at which point it is output to a list for subsequent manual

placement. Thus a situation may occur where the placement of one label with several available positions may preclude the placement of another label with only one position.

The first automated name placement system (Yoeli, 1972) performed in this manner (Fig 4.9). Large areal labels are placed first, and if these potentially overlap point features or other labels, then the area label letters are moved. This was then followed by the placement of small area and point labels. Basoglu's system appears to be an enhanced version of Yoeli's system but includes line placement and has a sophisticated name selection process (Fig 4.10). Another important improvement is that during point name placement, names are placed in order of the pre-defined importance of a point concerned. Thus as a general rule, it is only the least important names that are at risk from requiring manual placement.

In practice, although non-iterative placement strategies are simple to perform and consequently fast, the likeliness of overplotting and the requirement for manual editing to correct this suggests that these are semi-automated systems.

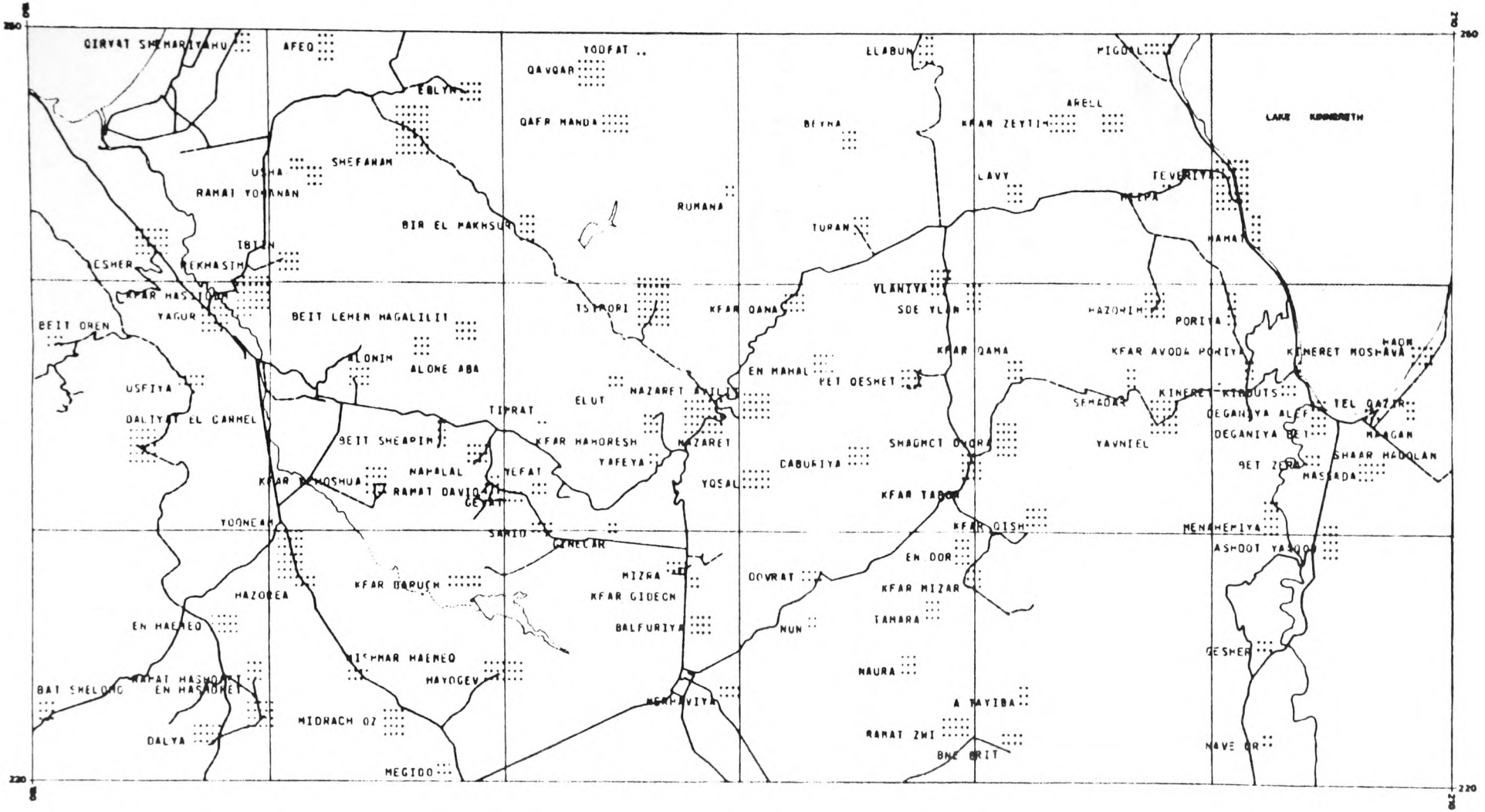


Fig 4.9 Yoeli's name placement (1972).

Fig 4.10 Best available reproduction of Basoglu's name placement (1982).

4.5.3 ITERATIVE

This usually involves each name being placed in its most preferred position and if any labels overlap, these are registered and a second pass is made to resolve these conflicts by moving the names to new positions. If overlaps still remain, further passes are made and the process repeats until all names have been successfully placed. However, because an endless loop situation may sometimes be encountered, most iterative methods have a cut off point which is triggered when a particular label has been placed a user specified number of times. To halt an endless loop of conflicts situation a label can either be deleted or fixed in position.

Hirsch's strategy is the only one which makes use of knowledge of which way to move overlapping labels to avoid overlap. After an initial placement, a map sweep is made every iteration to detect all overlaps starting with the lowest label in the sorted list of map objects (Section 4.4) and ending with the highest. At the end of each map sweep, labels are moved if they are found to be in overlap and the coordinates of each newly positioned label are stored and the list of objects is resorted. The process repeats until no labels remain in overlap.

On encountering an overlap(s), a "vector" is computed and represents a distance and direction needed to resolve

the conflict. In practice, because names are constrained to move around in circles, the vector mainly acts as a directional indicator. Hirsch uses two label movement methods to solve a conflict. The first method involves moving labels in the vector direction with sequential steps at preferred positions around the radius of proximity circle. This generally moves a conflicting label away from the denser areas of the map [Rule 2.74]. However, should this fail to resolve a conflict after several iterations, a second method is applied which involves the label in discrete jumps to new positions indicated by the vector angle. This ensures that such labels are shuffled in position and stand a better chance of conflict resolution when the first method is attempted again.

In Langran & Poiker's system (Fig 4.11), after point names have been selected, label positions which overlap the map border or other features are rejected. These overlaps are recorded to avoid future use of these combinations of positions. Each label is then placed in its best position. Next overlaps are found for each label pair and steps taken to avoid this in the following order:

1) Shift one label west if possible.

2) Shift one label north if possible.

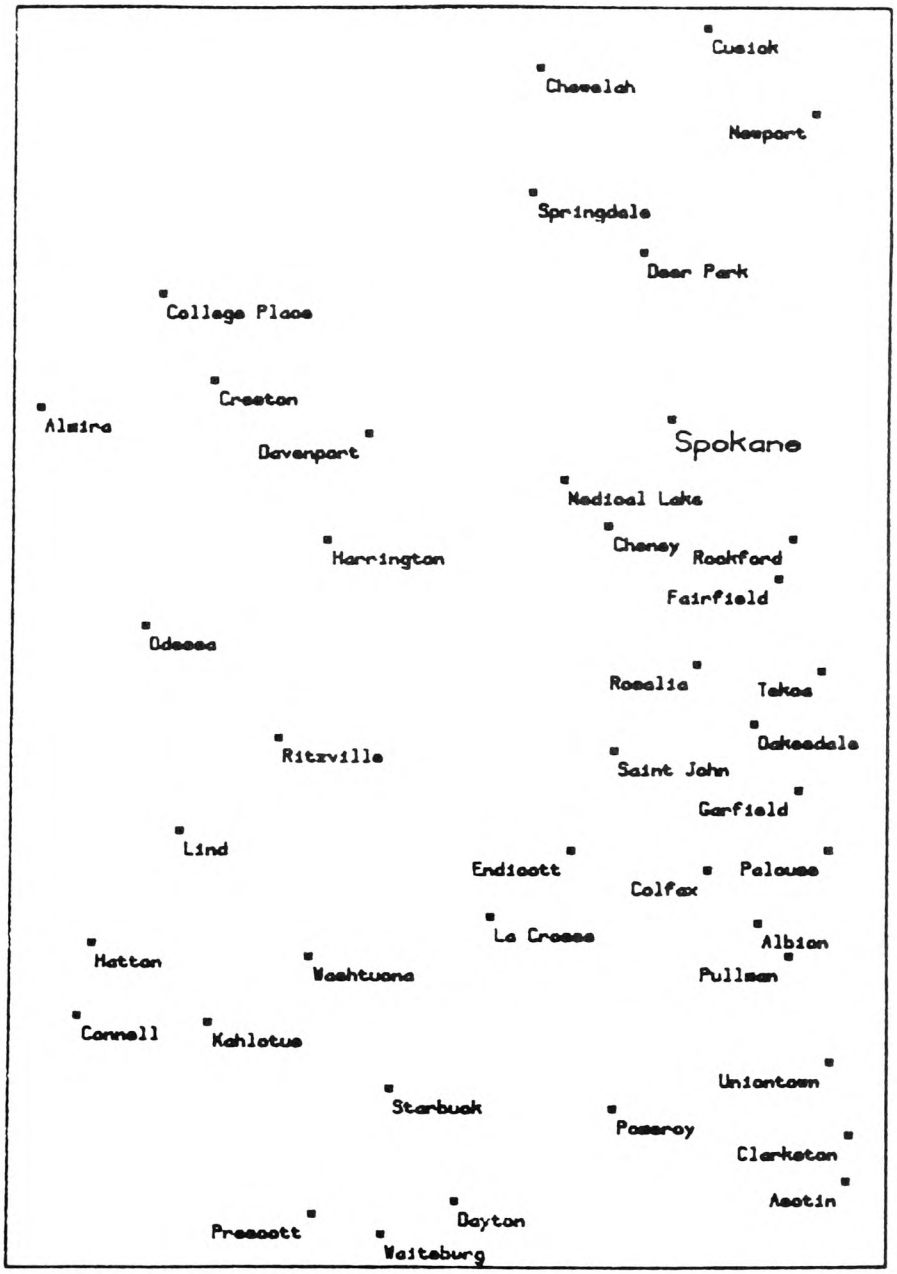


Fig 4.11 PLACENAMES name placement (Langran and Poiker, 1986).

3) If one label has the most unoccupied positions, shift this to its best available position.

4) Shift the least important label to its best available position.

If all of these fail, then all labels blocking the current label must be examined to see which one can be deleted. The least important label and its point feature are deleted. Once a label has been deleted then the labels are resorted and the placement iteration starts again. The iterations repeat until a solution is found.

4.5.4 BACKTRACKING

Backtracking involves the ability to undo placements if these lead to conflict and alternative combinations of placement exist. This is usually performed by keeping a record, on a stack, of which combination of positions have been used.

Greggains (1982) introduces the idea of backtracking in his AUTOTEXT system which uses the following strategy:

1) Select a theme for placement (Point, line or area).

2) Find label conflict clusters which are independent of each other.

3) Process names in each cluster by placing names with maximum potential conflict prior to labels of lesser potential conflict.

4) For each label, all placement positions which lead to irreconcilable overlap with underlying features, determined from the pixel map, are rejected and those that remain have their undesirability index values computed with respect to any acceptable underlying detail. The positions are then sorted into an order of preference.

5) Find potential conflicts between labels at certain positions and enter these into the conflict table.

6) Find a set of placement positions free of conflicts using a recursive backtracking procedure.

7) Enter placed labels into pixel map.

8) Repeat for all other themes.

9) Place names manually for which no solution was found.

AUTOTEXT orders the names for placement so as to minimise chances of future conflicts. It finds the name with the maximum number of conflicts, marks this as placed, orders its unmarked conflicting neighbours as described below, and selects them in this order. For these names in turn the same process is performed. Should placement at any level fail, then the conflict resolution process depends upon backtracking.

The desirability of label positions can be determined from the undesirability index value for a label at a given position by summing the pixels contained within the label's area. The pixels have weights which vary according to the object's importance. Thus by varying the weighting number the commitment to preferred orders of positions can be changed.

Unfortunately, no published results produced with Greggains system could be found so it is not possible to comment on the practicality of the system.

Mower (1986) also utilises backtracking strategy (Fig 4.12) which works through constraint propagation:

1) Order all point features according to degrees of freedom

2) Take the current feature

3) IF not encountered before THEN
   GOTO step 4
   ELSE
   GOTO step 7
   ENDIF

4) Place current feature label at best position

5) If current label overlaps another point, move the label to the next available position. Repeat step 5

6) If no success in finding positions for current label due to overlaps with points at all positions, then:

FOR every placement which overlaps features of less than or equal importance to the current feature, count the number of interfering points at these.

IF all the features overlapped are of a greater importance than the current feature, THEN

Delete the current feature and label

ELSE

Delete all features at the interfering placement and place the current label there.

ENDIF

NEXTFOR

GOTO 8

7) If the current label overlaps the current placement of a neighbouring object, update the placement of the feature of lesser importance. However if no more placements exist for the feature of lesser importance then delete it.

8) If a label has been placed for the first time or moved in position then place its neighbours at the top of the list of labels to be placed next.

Jones (1987) has developed a name placement system which utilises PROLOG's ability to backtrack to place point names on a map (Fig 4.13). Names are grouped into clusters of mutually overlapping labels and these are tackled separately. When placing a label, a check is first made to ensure that the placement will not overlap any existing labels or points, then a check is made to see if any underlying features are present. If either no underlying features are present or those that are have an importance which is below a user specified threshold value then the position is valid. If all positions tested are found not to be suitable then the system backtracks and unplaces the previous label and tries this at a new position. If the placement of a whole cluster of labels fails then the placement repeats but the threshold value is increased to allow for an improved chance of placement.

4.5.5 EXPERT SYSTEMS

Pfefferkorn et al (1985), introduce the ACES cartographic expert system for name placement (Fig 4.14). It is provided with a problem solver which searches

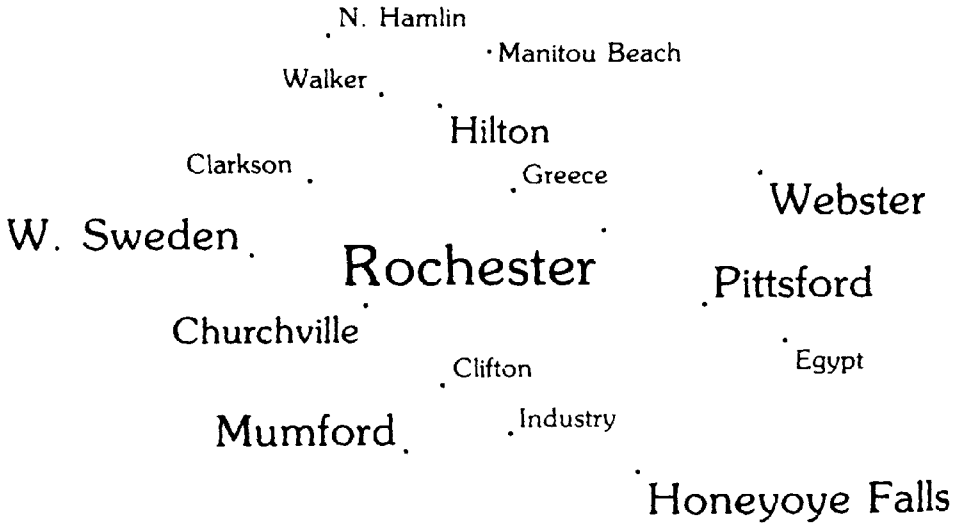


Fig 4.12 Mower's name placement (1986).

Fig 4.13 Jones' name placement (1987).

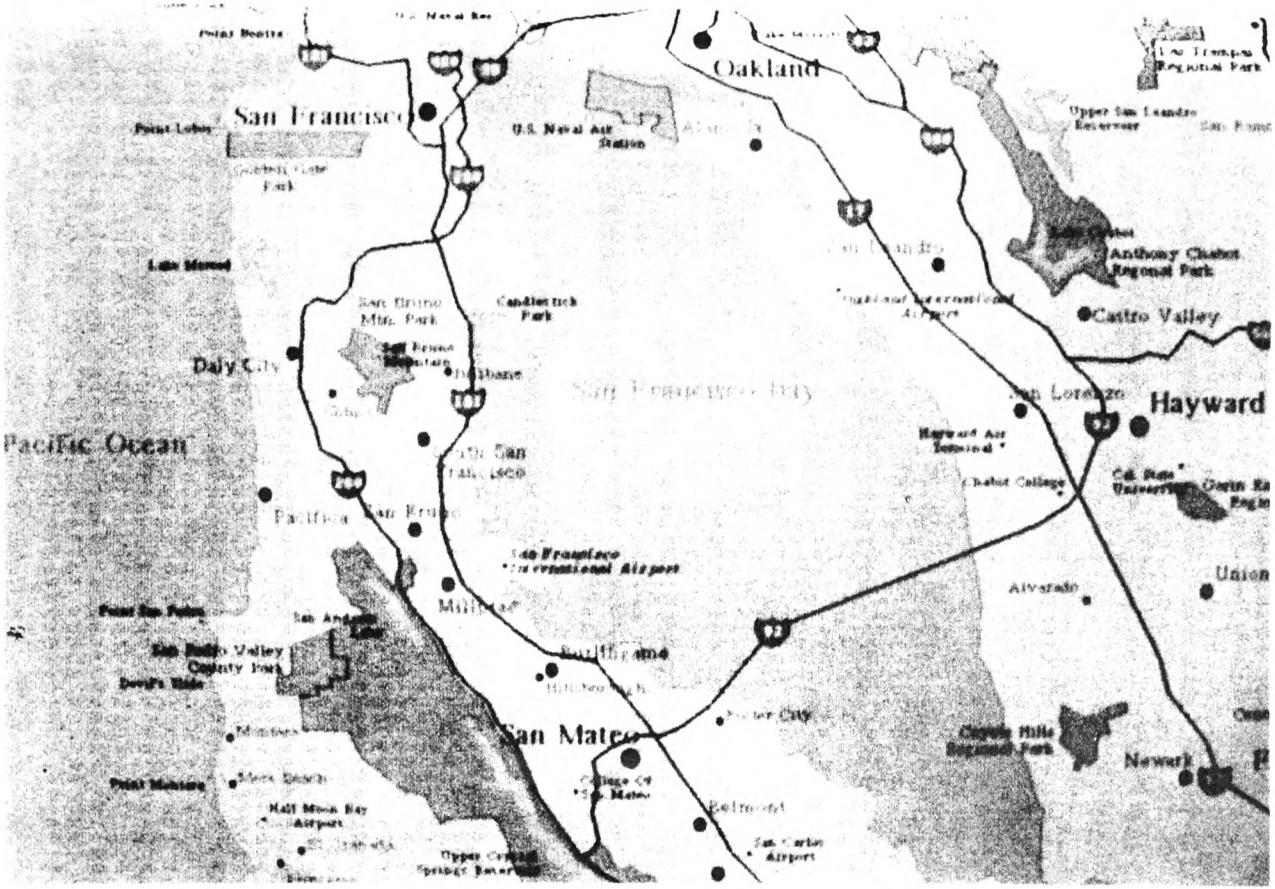


Fig 4.14 ACES name placement
(Pfefferkorn et al, 1985).

through a decision tree whose nodes are features to be labelled and whose branches correspond to strategies to perform the labelling.

In ACES the knowledge representation is split into a labelled feature details, an interaction graph and a decision tree. The interaction graph contains the influence rectangles of the labels for potential overlap detection. The decision tree controls the search for a name placement solution and records the strategies which have failed in the past.

To start the system an initialisation process must take place to compute possible label positions, potential overlaps and class priority. ACES then performs label placement iteratively.

Freeman and Ahn have developed AUTONAP, a rule-based system which utilises the knowledge of cartographers and general name placement considerations (Fig 4.15). They state that the rules in the system do not form a closed set and so the extension of the software and rule-base are catered for. In their rule-base there are a main set of rules which relate to all three label types, such as those discussed in chapter two, and secondary rules which refer to the usage of the primary rules where one rule conflicts with another.

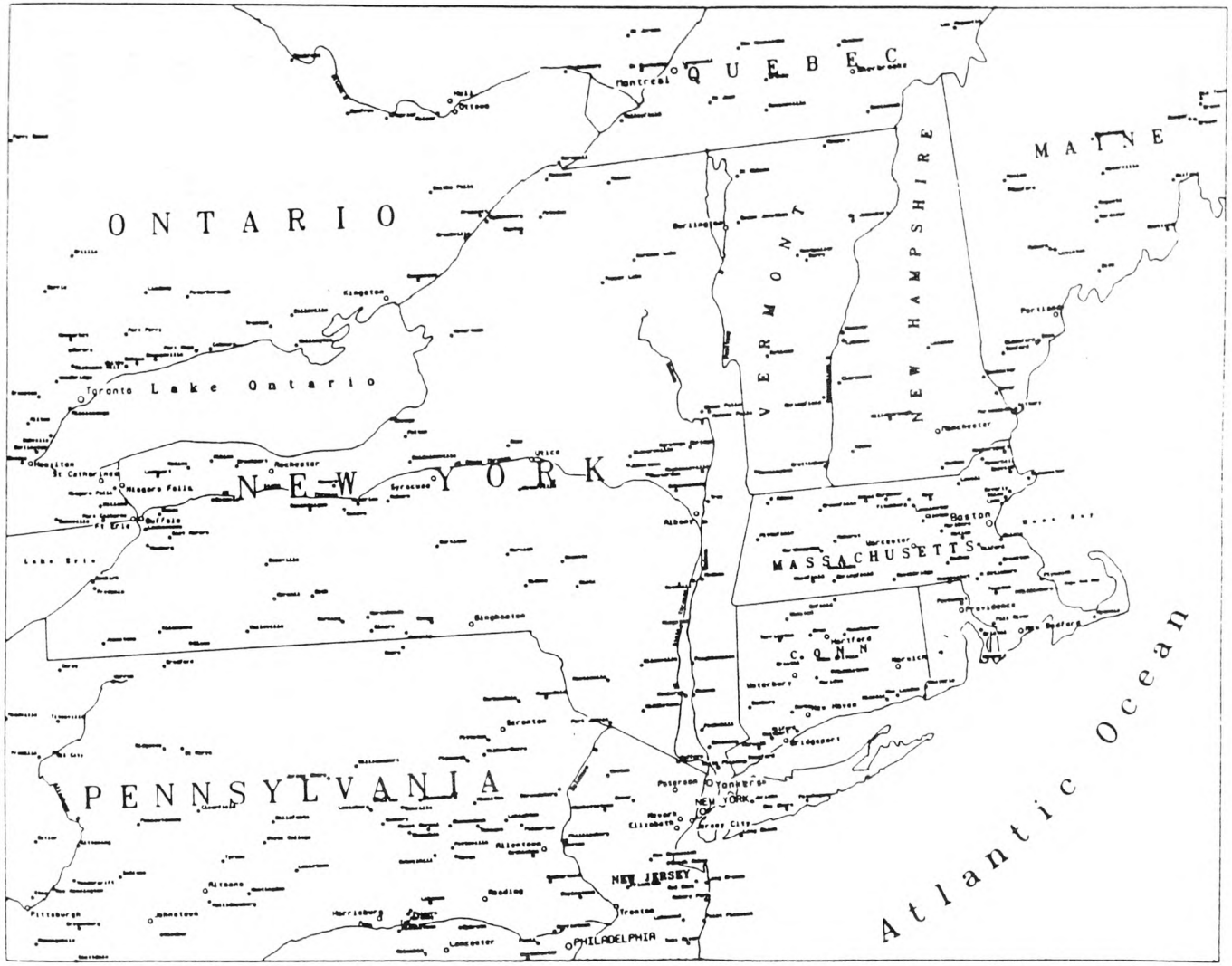


Fig 4.15 AUTONAP name placement (Freeman, 1985).

When placing point names, a graph of possible point name positions is generated. Then the graph is divided into connected components such that each node in the graph corresponds to a point feature and these are connected if name positions can potentially overlap. Clusters of potentially overlapping point names are then determined by means of a breadth-first search of connected nodes. Each cluster of potentially overlapping names is processed separately. For each node in a cluster of potentially overlapping labels, a list of free space grid cells is built.

Freeman and Ahn state that a "state space search" is made using the free space and a possible positions list using a heuristic graph searching algorithm similar to A*. The initial state is where no names have been placed and the goal is where all point names have been placed. The nodes are ordered so that the node with the smallest degree of freedom is checked first despite the positions of the nodes. This results in larger names being placed first. When a name cannot be placed, backtracking occurs and causes previously placed names to be placed at different positions. Backtracking at present does not go beyond separate label types.

Whilst it is fairly easy to get the system to avoid overlap and satisfy the basic requirements for name placement, Freeman and Ahn state that aesthetically

pleasing results were more difficult to achieve and several new rules had to be introduced to improve the aesthetic appearance of area placement.

4.5.6 TIMINGS

Very few authors state the time required to place names on their maps. The running time of Mower's system is on average 3 seconds for 44 names on a VAX 11/780 and the run time is predicted to rise linearly with the number of labels. Jones (1987) claims name placement timings of between 35 and 620 seconds, depending upon the map and name placement rules used, for about 50 labels on a VAX 11/785. Zoraster (1986) claims to be able to place as many as 2100 names in one hour on a VAX (Fig 4.16).

Unfortunately all these values are difficult to compare because the maps are of different densities, cartographic data access times are likely to vary, and the number of name positions used differ between authors. Also, no mention is made of whether these timings refer to the whole duration of programs or just the name placement component.

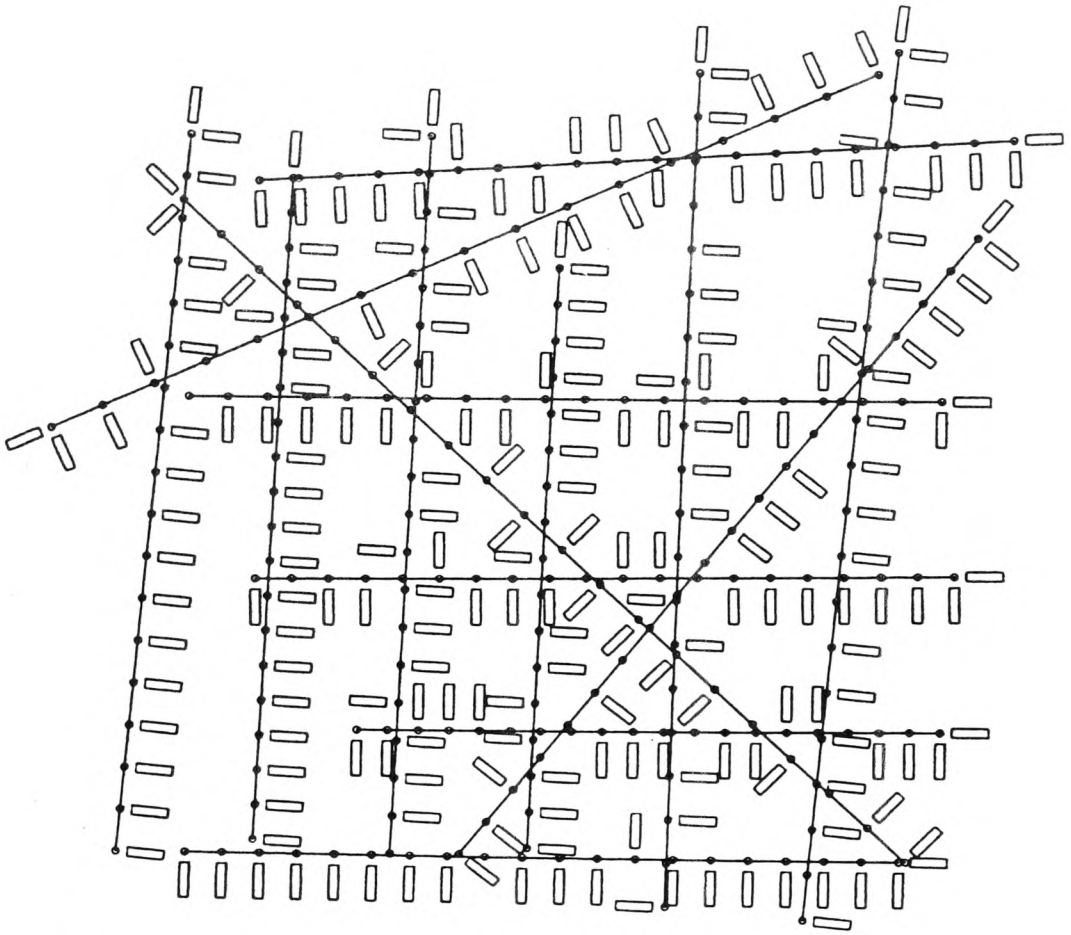


Fig 4.16 Zoraster's name placement (1986).

4.6 CHAPTER DISCUSSION

Although this author's research does not cover name selection to any great extent and instead mainly relies upon the features to label already being supplied in a pre-selected form, name selection is an important issue for any future automated name placement system to consider. For this reason, all information required to perform name selection must be available in the cartographic database.

With regard to name positioning algorithms, undoubtedly the most reasonable way to place point names is with the radius of proximity method. The number of positions used depends upon the user's requirements, for instance as few as four are suitable for crude but fast placement, but perhaps as many as twenty are required to satisfy the most finicky of cartographic specifications. Similarly equispaced positions along a line could be used for placing a line label on, and if as many as twenty were used this should be enough to allow a sufficient number of positions to choose from even though it may be decided to reject some of these because they lie on "curved" sections of the line. Area placement is less easy to parameterize and will be considered more carefully in the next chapter.

Many automated name placement systems detect overlaps between labels and other labels or underlying details with

the use of raster data. Nevertheless detection of overlaps between labels may be accomplished more easily by the use of point in polygon type tests such as those described by (Hirsch, 1982). As an optimum solution, raster data will be used for detecting overlaps with underlying features and point in polygon type tests applied to label boundaries for detecting overlaps between labels.

With regard to name placement strategies, it is clear that there are almost as many strategies as there are name placement systems. Of these the non-iterative strategies are no longer worthy of investigation and the author's research will be directed towards implementing new iterative and backtracking strategies.

Hirsch's (1982) conflict resolution method, of computing vectors which indicate the direction and extent to which a label should be moved to resolve overlap, has the most similarities with the way human cartographers resolve label overlaps. However according to Basoglu (1984) it requires 15 iterations to place names on a relatively sparse map. Most other name placement algorithms attempt to place names in a set of predefined positions until a suitable position is found. These also tend to place area names prior to point names prior to line names, however this presupposes that area names are the most difficult names to place, but this is not always the case. Although Greggains adopts a similar approach, he

suggests placing labels with the maximum potential conflict first. An acceptable alternative, in the author's view, would be to place those with the least number of positions first because these could also be regarded as the most difficult to place.

Langran and Poiker (1986) reduce the choice of point name placement positions to just four since they claim that these resolve most label overlap problems, but this will lead to a reduction in the aesthetic appeal of their point label placements. They also apply an extensive label selection criterion which appears to reduce the labels density to such a low value that overlaps between labels are quite unlikely to occur. Nevertheless their iterative method of placing labels appears quite sensible (Section 4.5.3) and is designed to run on very small systems.

Backtracking name placement methods as described by Greggains (1982), Mower (1986), Jones (1987) and Freeman and Ahn (1984) produce acceptable results, but not much attention has been made to avoiding ambiguity.

A name placement technique not covered in this review is the use linear programming (Section 4.1). Cromley (1986, see Fig 4.17) was first to apply a modified form of linear programming to the name placement problem where the aim is to "minimise the sum of the priority weights for the entire label distribution". This will find an optimal

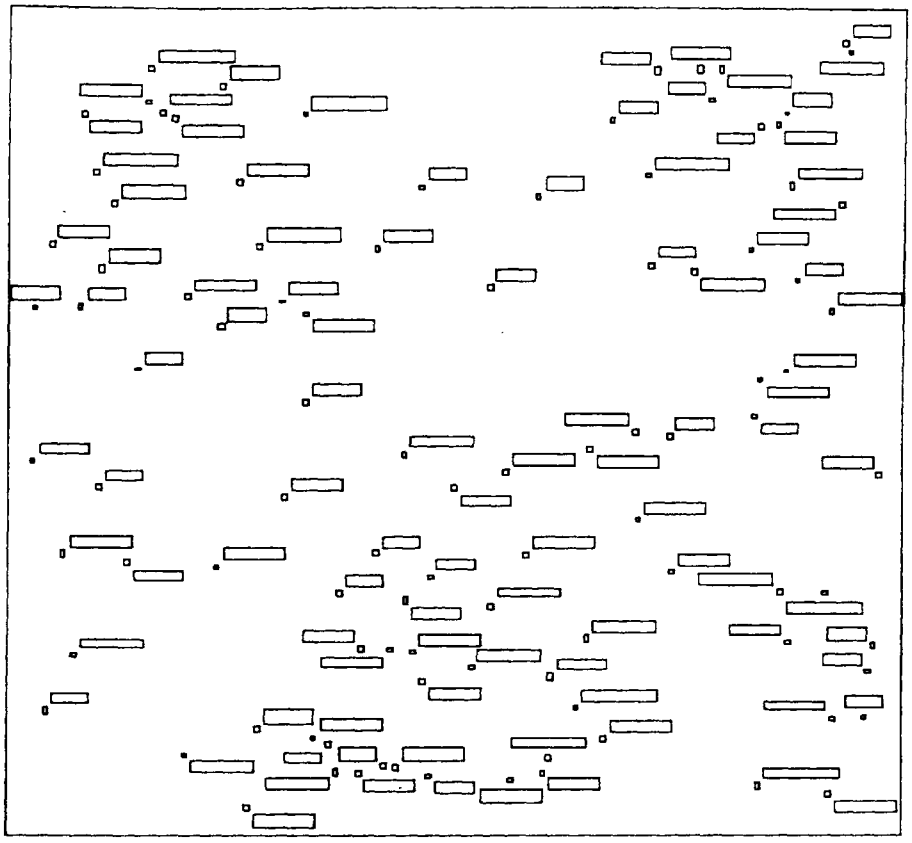


Fig 4.17 Cromley's name placement (1986).

solution if one exists, however if not then a relaxation procedure is used.

Finally, although just over a dozen automated name placement systems have been published, there are several fundamental problems which remain and these have been investigated in the author's research:

1) In order to implement automated name placement systems, name placement rules, as used by expert cartographers, are required. In the published accounts of automated name placement systems, studied by the author, little information is available on how name placement rules are extracted apart from using previously published rules such as those used by Imhof (1975). The author has had only a limited success at extracting rules from cartographers despite the use of map analysis techniques (Chapter 2).

2) Label position algorithms and label configurations used in previous automated name placement systems are not sufficient to cater for the variety of those used on paper maps. The author's research has included a more extensive investigation of how to implement different label configurations (Chapter 5) than have previously been published.

3) Ways of reducing label ambiguity used by previous researchers have either been ineffective or restrictive. This problem is addressed in Chapter 6.

4) Map design and optimization of name placement efficiency should be investigated because, to the author's knowledge, no previously published accounts of this exist. These problems are addressed in Chapter 6.

5) A means must be found of making the inclusion of rules into an automated name placement system easier. Chapter 8 demonstrates how logic programming can be utilized for the implementation of name placement rules at a higher level (to the author's knowledge) than has been achieved previously.

6) Published name placement methods, such as Pfefferkorn et al (1985, see Fig 4.14), often include very few examples of automated name placement. The author has tried to demonstrate the flexibility of rule-based name placement systems for labelling different types of maps and with many varieties of map design (Chapter 8).

CHAPTER 5

POSITIONS, CONFIGURATIONS AND DIMENSIONS OF LABELS

5.1 INTRODUCTION

When developing a practical automated name placement system, one must allow for most of the essential positions and configurations used on a wide range of maps. Configuration is a general term which refers to the arrangement of a label with respect to its feature. The label configurations discussed in this chapter are based mainly upon the author's study of label configurations in chapter 2 and also upon some of the name placement techniques reviewed in chapter 4. Any cases not considered here will be due to their rarity, complexity, or because suitable alternatives exist.

As discussed in chapter two, names can be placed in an infinite number of locations. However in performing the task on a computer, the positions of labels must be quantised into a finite number of possibilities. Twenty was chosen as a reasonable maximum number of positions to consider for each label in any one configuration. On some occasions, when it may not be possible to place a label at any of its positions, the configuration or dimensions of the label can be changed to find a solution and avoid the

need for label deletion.

The means of storing the positions of horizontal, diagonal or curved, labels depends upon the label configuration. Horizontal label positions can be stored using just the eastings and northings of the centre of each label. Diagonal label positions can also be stored using the coordinates of the centre of the label, but need an angle to be specified in degrees. To define the positions of curved names requires the mathematical description of the curve along which the label lies and the position of the centre of the label along the curve. The curved labels considered follow circular arcs, parabolic and cubic curves.

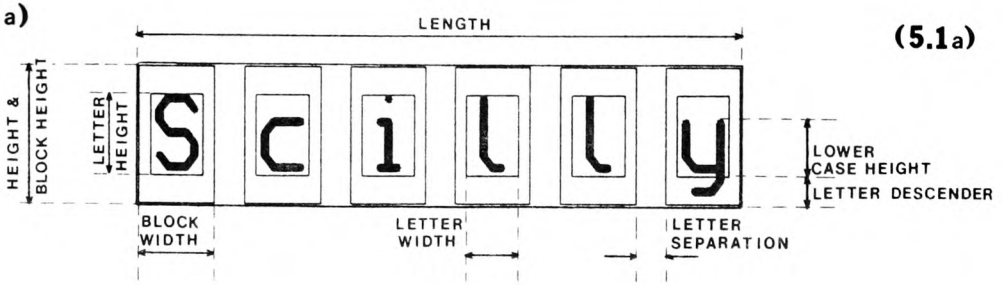
In addition to the above configurations, labels sometimes need to be presented with letters or words spaced out and sometimes split onto separate lines. Fig 5.1 a, b and c show how the dimensions for these three different label configurations are found for non-curved labels. The letters making up a label must all be of the same print font character size. For computing purposes a font character size is defined by a character block height and width. Inside the character block resides the character letter of predefined height and width. The character height and width should be smaller than the corresponding block size so that, if character blocks are placed next to each other without a gap, the letters will

Height = Block height

Length = No. of letters x Block width

+ (No. of letters - 1) x Letter separation

(5 1a)

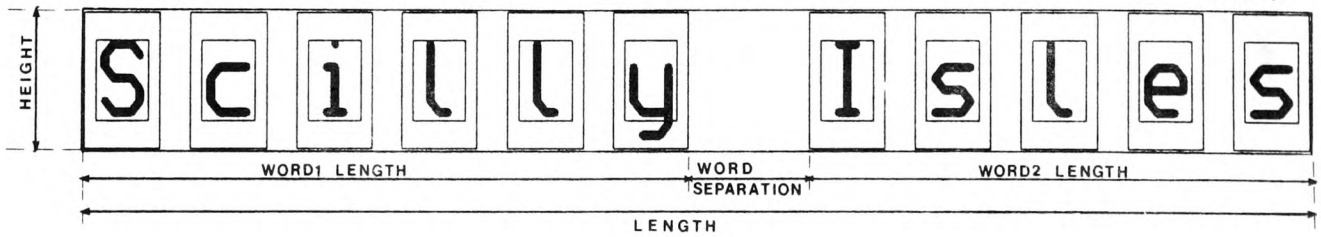


(5.1a)

(5.1b)

Height = Block height

$$\text{Length} = \sum_{i=1}^{\text{No. of words}} \text{Word}_i \text{length} + (\text{No. of words} - 1) \times \text{word separation}$$



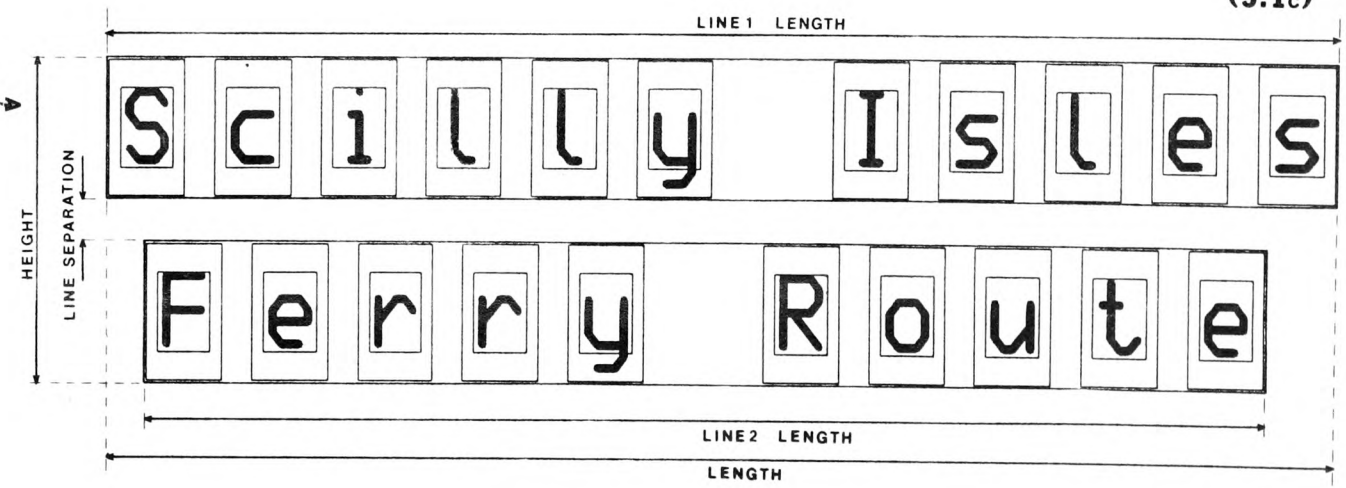
(5.1b)

(5.1c)

Height = No. of lines x Block height

+ (No. of lines - 1) x Line separation

Length = maximum label (line) length



(5.1c)

Fig **5.1a** Dimensions of a single word label.

Fig **5.1b** Dimensions of a multi-worded, single line label.

Fig **5.1c** Dimensions of a multi-lined label.

not touch. Each letter is offset from the bottom of each block by a letter descender height which allows for the minority of letters with descenders, namely "g,p,q and y". The placement of a character inside a character block is immaterial to name placement but ideally should be centralised. Additionally, letter spacing, word spacing, and line spacing should be catered for to allow for labels to be spread out without changing the specifications of the character block contents. All character font parameters associated with different labelled features should be defined according to the style of the map. In all cases of non-curved labels, the centre of the label will be deemed to be at the centre of a rectangle bounding the character blocks.

The irregular spacing of letters will not be considered. This rarely occurs except in the case of atlas maps and in situations of high label density. In both cases, alternatives exist such as reducing letter size, increasing letter spacing, or using some other configuration.

To demonstrate the principles of an automated name placement system, it is only necessary to implement a few of the label configurations discussed in this chapter.

5.2 POINT NAME PLACEMENT

5.2.1 INTRODUCTION

Three principal configurations of point label placement were found on the wide variety of maps discussed in chapter two. These include the horizontal placement of a label around and at a fixed distance from the point. Secondly, arrowed labels which are used in situations where it is not possible to place the label close to its point. Thirdly, diagonal and curved labels which are rare, but do occur on some atlas maps, especially in the vicinity of coastal regions. Although this section will consider all three types, only horizontally placed point labels will actually be implemented.

All labels, unless curved, will be treated as rectangular boundaries encompassing the letters within the label. When placing a label, the position deduced will refer to the centre of the label bounding rectangle (Fig 5.1).

In general, most point labels are of the horizontal type. So, given details about the dimensions of such a label, only the label position and the shortest distance between the point and the label edge, known as the radius of proximity, need be defined to place the label relative to its point feature uniquely.

5.2.2 HORIZONTAL POINT NAME PLACEMENT [IMPLEMENTED]

Given that a label is of length L, height H and radius of proximity R, then for a continuous range of positions the centre of the label describes the locus shown in Fig 5.2. The locus consists of a rectangle of length L+2.R and height H+2.H with its corners truncated by arcs of 90 degrees and of radii R.

Fig 5.3 shows each of the twenty possible positions defined for the label. It can be seen that positions 2, 4, 8, 10, 12, 14, 18, and 20 have been selected to lie at the change over points between the straight and curved sections, and positions 1, 6, 11, 16 and 3, 9, 13, 19 lie in the middle of the straight and curved sections of the locus. Because labels tend to be of considerable length as compared to their width, it is necessary to introduce quartile positions 5, 7, 15, and 17 along the top and bottom straight sections of the locus in order to allow for a more continuous distribution of positions.

The equations given in table 5.1 can be used as a look-up table in a name placement algorithm enabling the label's relative position from the point DX,DY to be found and added to the point coordinates PX,PY, to give absolute label coordinates.

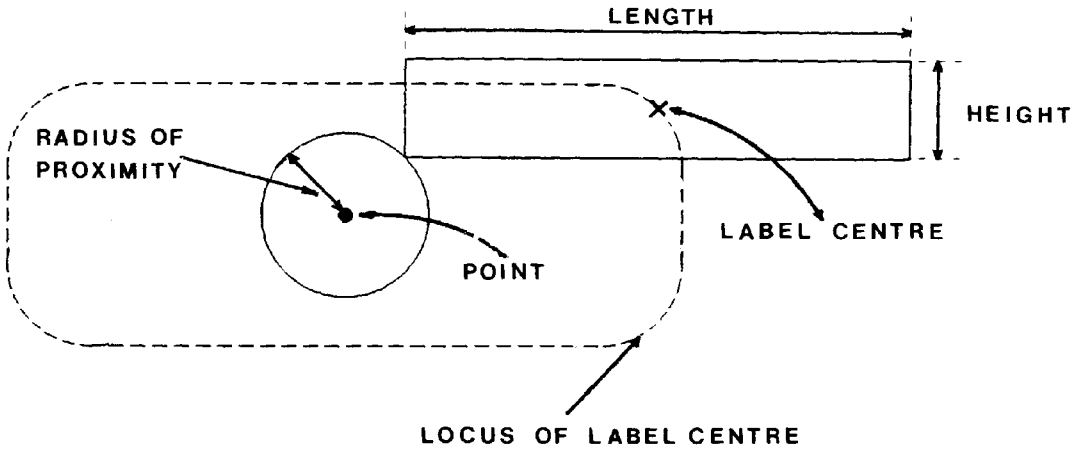


Fig 5.2 Point label dimensions.

Table 5.1 Relative offsets for label from a point at different positions. Radius of proximity R, label length L, label height H.

Label Position Number	DX Displacement from Point OX	DY Displacement from Point OY
1	R+L/2	0
2	R+L/2	H/2
3	$R/\sqrt{2}$+L/2	$R/\sqrt{2}$+H/2
4	L/2	R+H/2
5	L/4	R+H/2
6	0	R+H/2
7	-L/4	R+H/2
8	-L/2	R+H/2
9	$-R/\sqrt{2}$-L/2	$R/\sqrt{2}$+H/2
10	-R-L/2	H/2
11	-R-L/2	0
12	-R-L/2	-H/2
13	$-R/\sqrt{2}$-L/2	$-R/\sqrt{2}$-H/2
14	-L/2	-R-H/2
15	-L/4	-R-H/2
16	0	-R-H/2
17	L/4	-R-H/2
18	L/4	-R-H/2
19	$R/\sqrt{2}$+L/2	$-R/\sqrt{2}$-H/2
20	R+L/2	-H/2

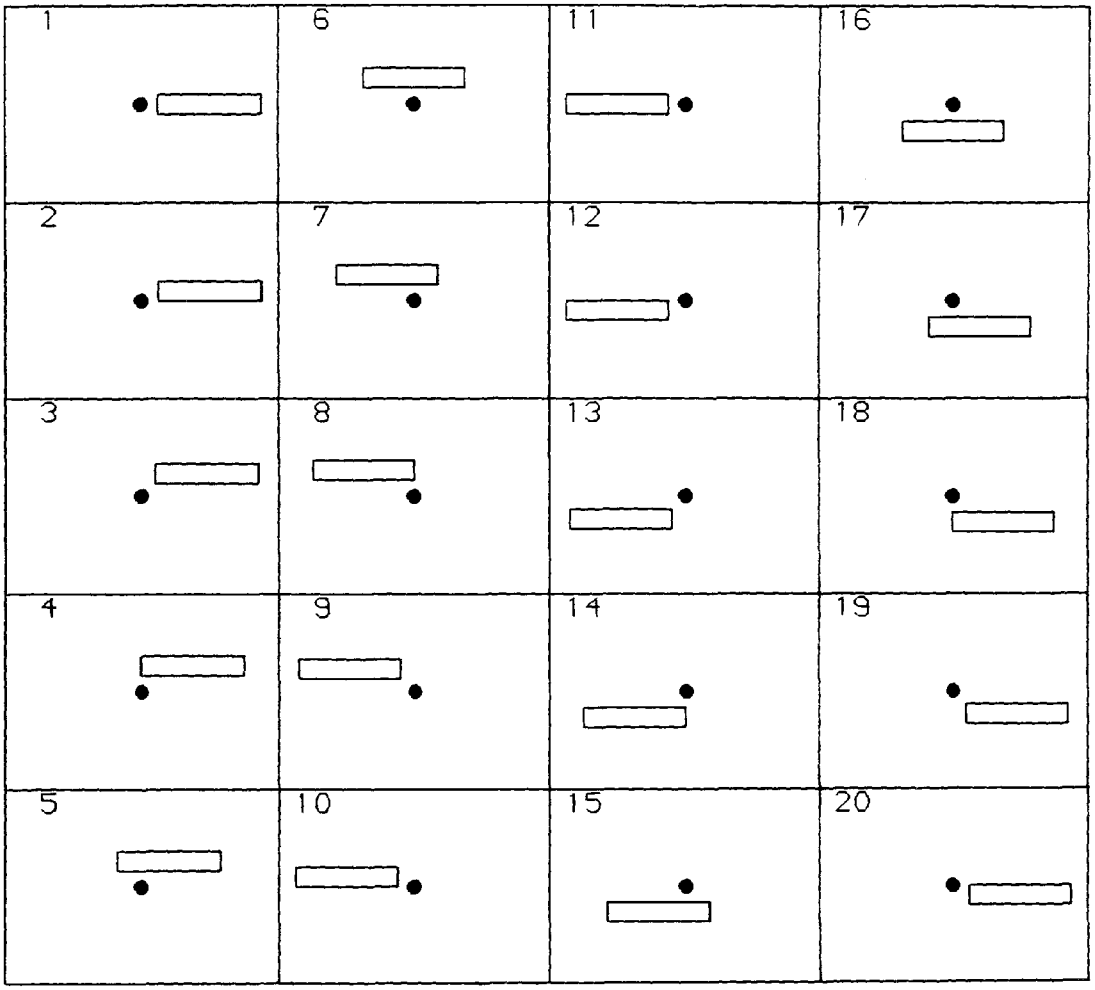


Fig 5.3 Point label positions 1 to 20.

5.2.3 ARROWED POINT NAME PLACEMENT [NOT IMPLEMENTED]

This type of name placement usually occurs in feature-dense regions of the map, or where the name cannot be placed next to its point because it is too long to fit into a confined space. Like the horizontal point name placement, a fixed radius of proximity could be used for a particular label. However, the radius of proximity and the selection of positions are defined differently. The reason for this is illustrated in Fig 5.4, which shows the irregular spacing of label positions that results using a large radius of proximity with the horizontal point name placement method. Instead, the placement of label positions are at fixed intervals of 18 degrees around a circle of radius of proximity R (Fig 5.5). R in this case is different and is defined as the separation distance between the point and the centre of the label. Because of the way the radius of proximity is defined it must be greater than L/2 otherwise the label will encroach the point.

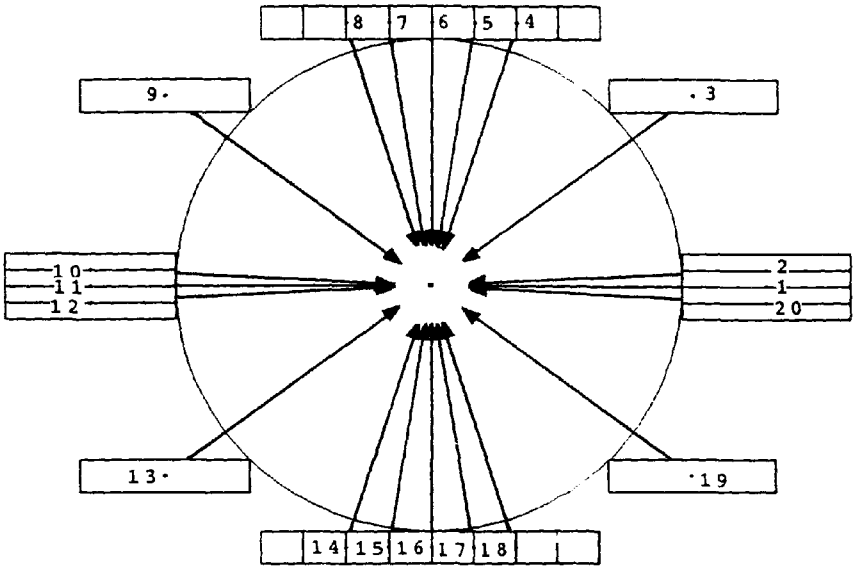


Fig 5.4 Point (arrowed) label placement with a large radius of proximity.

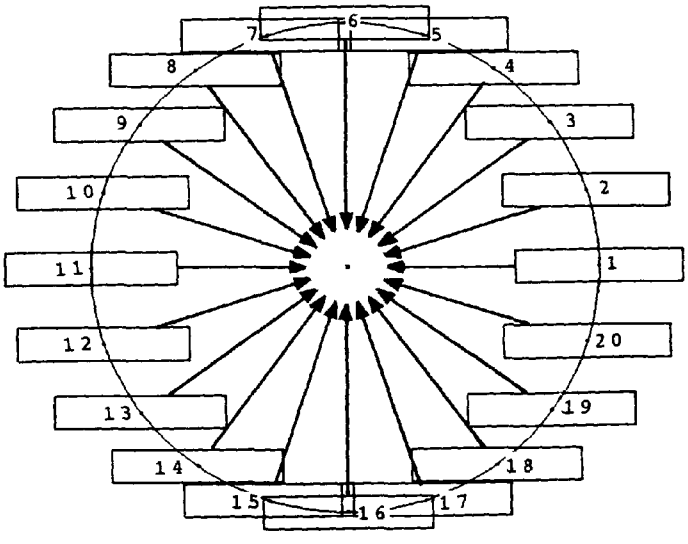


Fig 5.5 Arrowed point label placement.

5.2.4 DIAGONAL POINT NAME PLACEMENT [NOT IMPLEMENTED]

This type of placement is used on atlas maps along the coast where it is necessary to place point names at an angle to the coast. This can exist in two forms, one being identical to horizontal point placement except that the label is tilted through an angle (Fig 5.6). Secondly, as a radial distribution of names radiating away from the point and spaced at equal angles of 18 degrees (Fig 5.7).

5.2.5 CURVED POINT NAME PLACEMENT [NOT IMPLEMENTED]

This configuration is used with similar frequency to diagonal point name placement. Unlike diagonal point name placement though, only radial placement is considered. The type of curve used is a circular arc which seems applicable to most curved point names encountered by the author. Fig 5.8. shows the parameters needed to describe this kind of name placement. Note that the curvature of the label starts perpendicularly to the radius of proximity circle and that the label can curve two ways, clockwise or anti-clockwise. The centre of the radius of curvature of the label lies a distance P along a tangent to the radius of proximity circle.

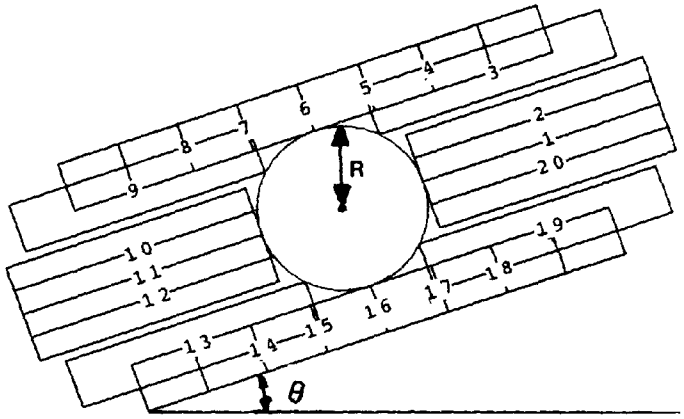


Fig 5.6 Parallel-diagonal point name placement.

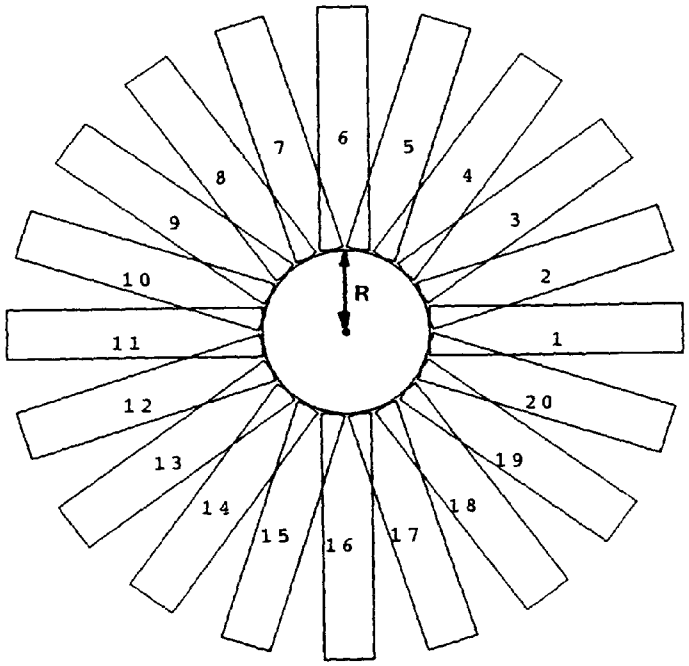


Fig 5.7 Parallel-radial point name placement.

5.2.6 SECTION DISCUSSION

Although only the horizontal point placement technique will be implemented, the point name placement techniques discussed here form a set of typical configurations found on maps.

Further possible developments to curved point label parameterization might be to allow for parabolic curves and an angle of projection of the label from the tangent of the radius of proximity circle. However it is the opinion of the author that these can be disregarded without either damaging the chances of successfully placing names or the aesthetic appeal of the map.

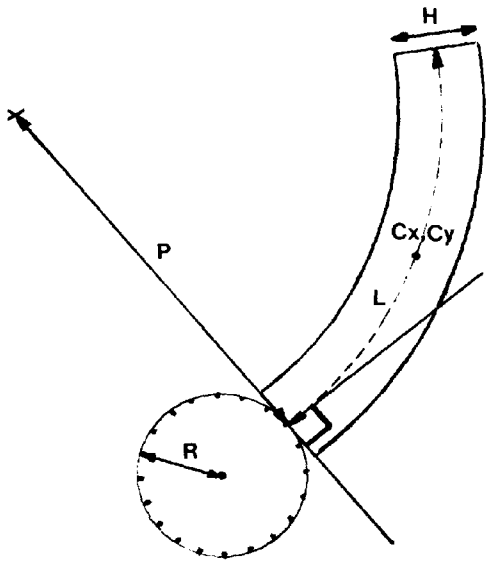


Fig 5.8 Curved point label placement. Label length L, height h, centred on (Cx,Cy), radius of proximity R, label radius of curvature P.

5.3 LINE NAME PLACEMENT

5.3.1 INTRODUCTION

Line labels fall into two categories, straight and curved. The shape of the line forms the basis of the locus for straight label positions and defines the curvature for curved labels.

5.3.2 STRAIGHT LINE LABELS INTRODUCTION

Straight labels can be placed in one of three configurations:

1) diagonal and parallel with the line.

2) horizontal across the line.

3) arrowed (e.g. see upper right corner of Fig 2.18).

For the first two configurations, twenty positions are used at regular intervals along the line, (Fig 5.9), the start (1) and end (20) positions lying at the end nodes of the line. Given the position number of the label, the coordinates of the label can be found by summing up the total length of links between adjacent coordinate

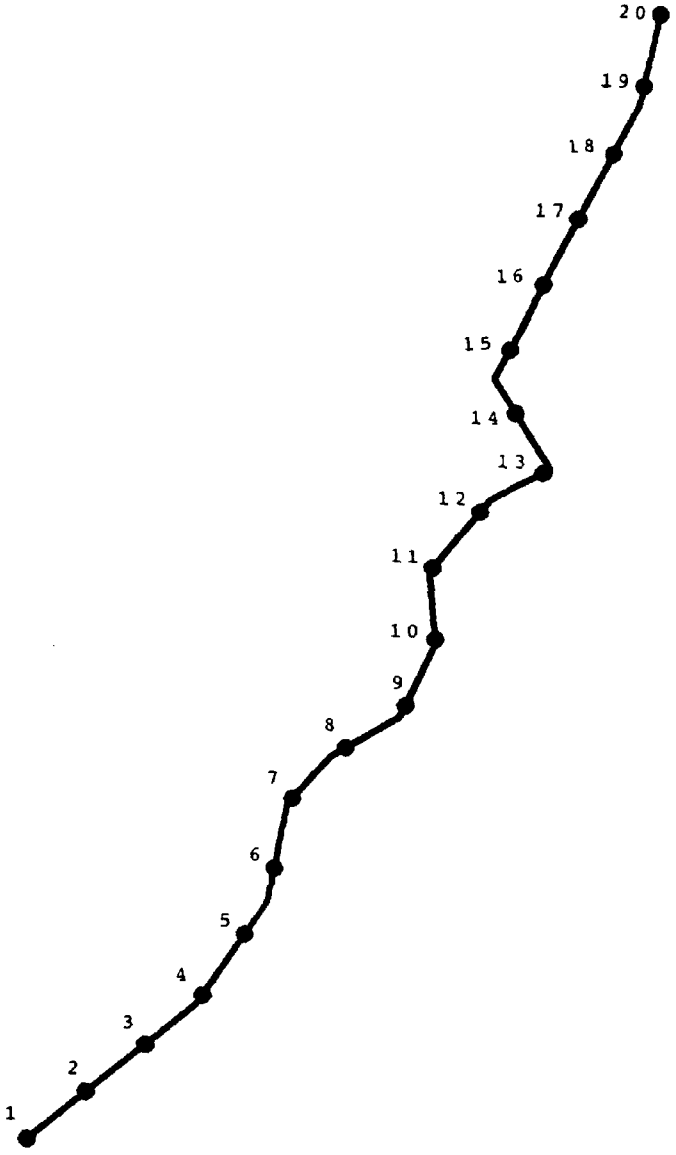


Fig 5.9 Twenty regularly spaced positions along a line.

pairs making up a line until the correct fraction of the line length has been traversed. Interpolation between adjacent coordinate pairs is used to find the precise location along the line corresponding to the specified position. The third configuration, "arrowed", is based upon arrowed point name placement.

5.3.2.1 DIAGONAL LINE LABELS [IMPLEMENTED]

Diagonal line labels generally lie parallel with and on top of the line, but can be offset above or below if required (Fig 5.10). When placing such labels, their angle of tilt is defined by constraining the two end points of the label to lie on the line (Fig 5.11), this ensures that irregularities along the line are averaged out over the label's length. When offsetting a diagonal line label, it is shifted along a direction perpendicular to the label tilt angle by half the label height plus the offset distance.

5.3.2.2 HORIZONTAL LINE LABELS [IMPLEMENTED]

Horizontal line labels are generally centred on the label position computed for the line. They are by far the easiest of the line labels to position because they are parallel to the majority of other labels on the map and

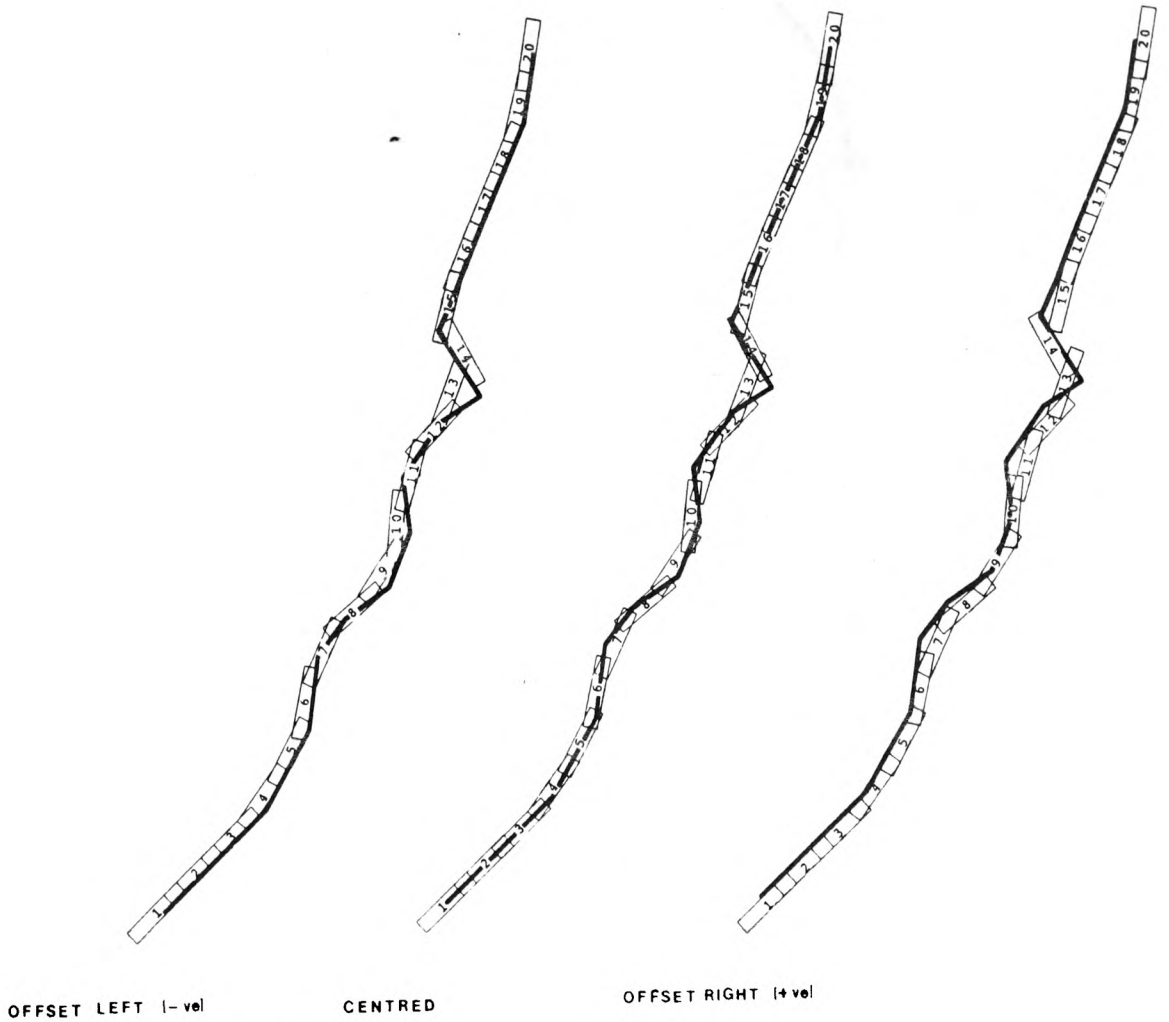


Fig 5.10 Twenty positions for diagonal line labels in each of 3 configurations.

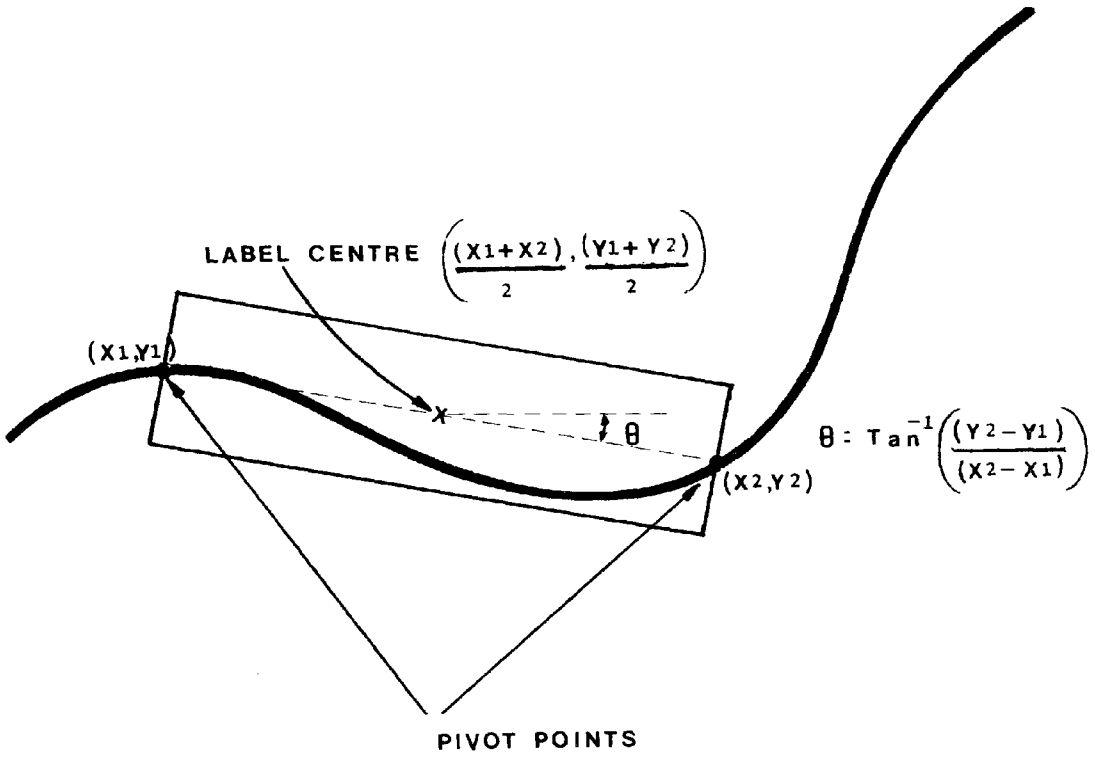


Fig 5.11 The pivoting of a diagonal line label about its two end points.

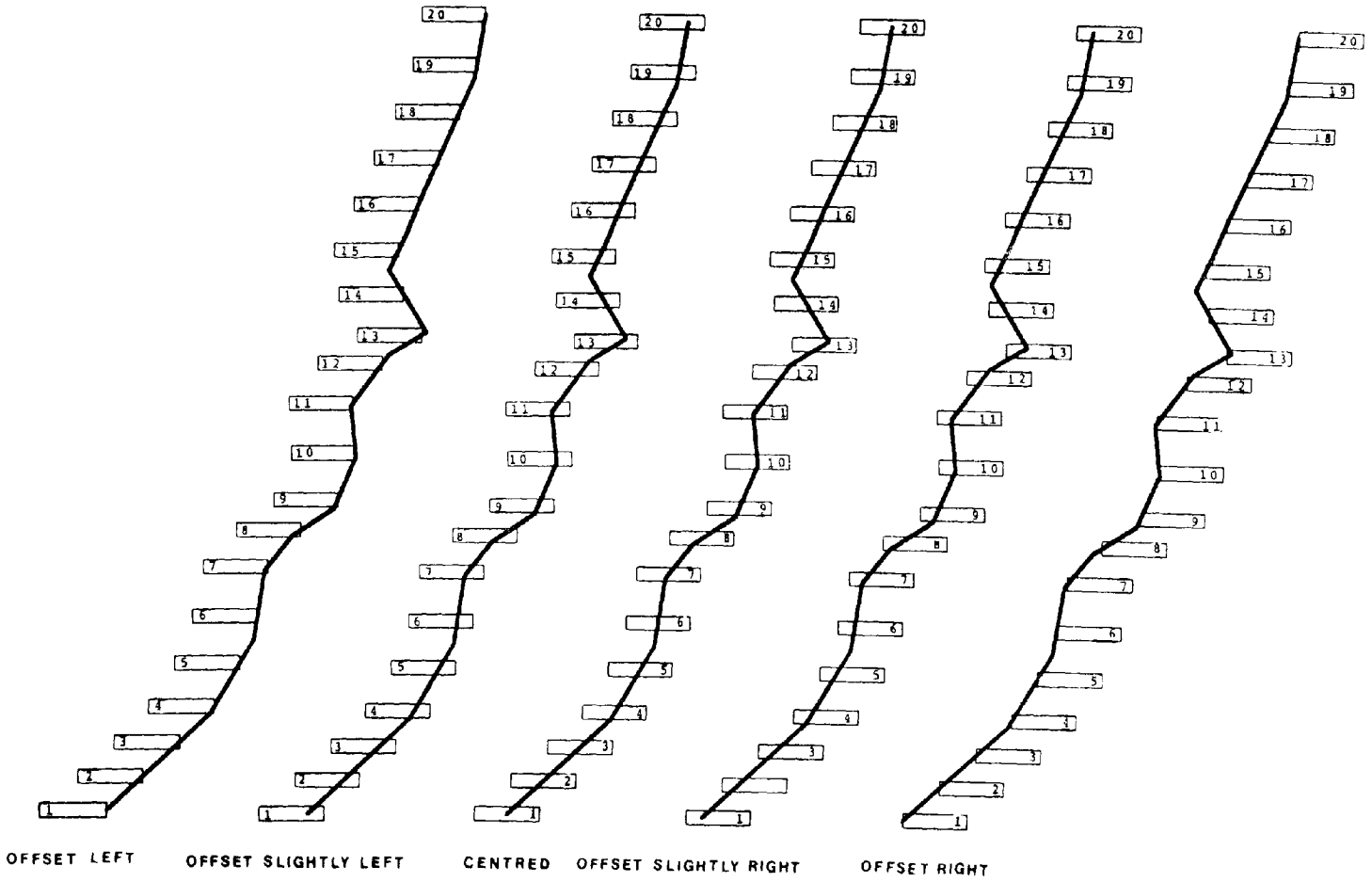


Fig 5.12 Twenty positions for horizontal line labels in each of 5 configurations.

their angle of tilt does not have to be computed since it is by definition zero. To offset a horizontal label, it is shifted by the appropriate amount in the easting direction (Fig 5.12).

5.3.2.3 ARROWED LINE LABELS [NOT IMPLEMENTED]

Arrowed line labels have such a wide range of positions, that it is not very practical to parameterize line label positions without stipulating where on the line the arrow points to. Arrowed line labels generally apply to very short sections of line, in fact when viewed from a distance these can be treated as point features. Thus one possible means of implementing arrowed line label configurations is to use arrowed point name placement but applied to the mid point of the line. Unfortunately label positions which lie along the direction of the line will result in the arrow overlapping the line (Fig 5.13). To avoid this such line label arrow positions must be tested so that those which obscure the line can be eliminated.

5.3.3 CURVED LINE LABELS [NOT IMPLEMENTED]

This form of line label configuration is often applied to features such as rivers where the label letters flow along the river length either above or below the

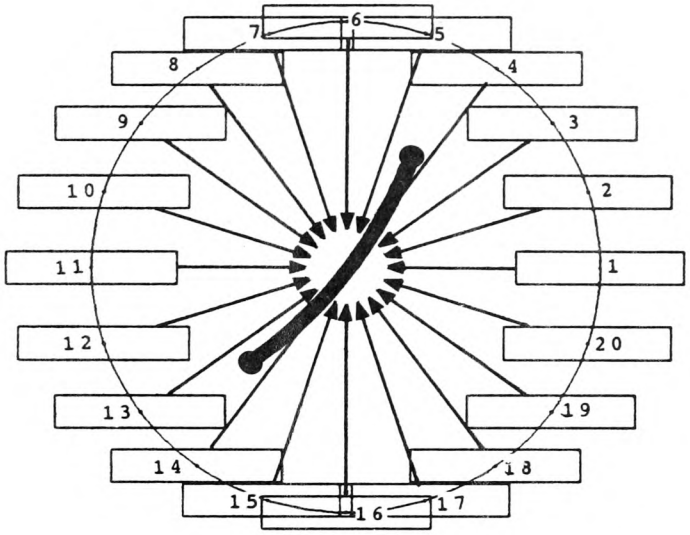


Fig 5.13 Twenty arrowed point label positions.

line. The letter spacing is usually related to the length and importance of the river, and there may be occasions in which the letter size is increased according to river importance. Curved labels should be placed on slowly varying circular arc, parabolic or "S" shaped cubic curves. Basoglu's placement algorithm is recommended for such curved labels (Section 4.3.3 and Fig 4.2).

5.3.4 SECTION DISCUSSION

Of the line labelling techniques discussed, only horizontal and diagonal line labels have been implemented by the author. This is because in order to demonstrate the principles of name placement only the most common placement configurations need be used.

To prevent line labels being placed over the end nodes of the line, the user should avoid using positions 1 and 20 and any others in the vicinity. This could be implemented by giving positions near the end of lines a much lower preference than others nearer the line centre.

Some of the line positions selected, may lie on wiggly sections of the line which are less preferred for placing labels on (Rule [2.30]). These positions could be detected by applying Sampson's method which is illustrated in Fig 4.3 (see Fig 4.3 and Davis and McCullagh, 1975).

5.4 AREA NAME PLACEMENT

5.4.1 INTRODUCTION

The most usual form of area labels are those placed wholly inside the area boundary. A technique was developed during the second year of research to place this type of label. It makes use of three separate algorithms for generating positions for horizontal, diagonal and curved area labels. Up to twenty positions can be generated for horizontal and diagonal area label placement algorithms, however the curved label algorithm is only capable of placing the label centrally inside the area either on a parabolic or cubic ("S" shaped) curve.

Since the range of area label positions and configurations is potentially very large, compared to those available to point or line labels, it is likely to take longer to find suitable positions. Because of this area label positions are computed prior to placement and up to twenty of the best of these are identified (according to a criterion outlined in section 5.4.3) and stored for use during placement. The disadvantage of using stored label positions, unlike point and line label positions which are computed when needed, is that should it be necessary to change the dimensions of the label then all the label positions will have to be computed again.

5.4.2 INITIALISATION

All three area label placement algorithms require a similar initialisation prior to determining positions for area labels. Firstly, all area data should be stored in a rasterized form with the minimum and maximum eastings and northings of its extremities determined. Using these limits, the maximum side of a rectangle bounding the area can be found (Fig 5.14). The area data is then transformed into a N x N array by sub-sampling the original area raster data and scaling it so that N pixels correspond to the maximum side of the bounding rectangle. Next, the centroid coordinates, calculated using pixels, and the angle of the maximum moment of inertia (major axis) of the area in the array are found.

If the area label is to be placed in a diagonal or curved configuration then the array containing the area must be rotated through the angle of the maximum moment of inertia about the centroid, and into a larger 3N x 3N array, so that the major axis of the area is horizontal (Fig 5.15). This helps with the computation of positions for diagonal and curved configurations where the label generally lies along the major axis of the area. The use of an array three times bigger than the original avoids any pixels being rotated outside the array.

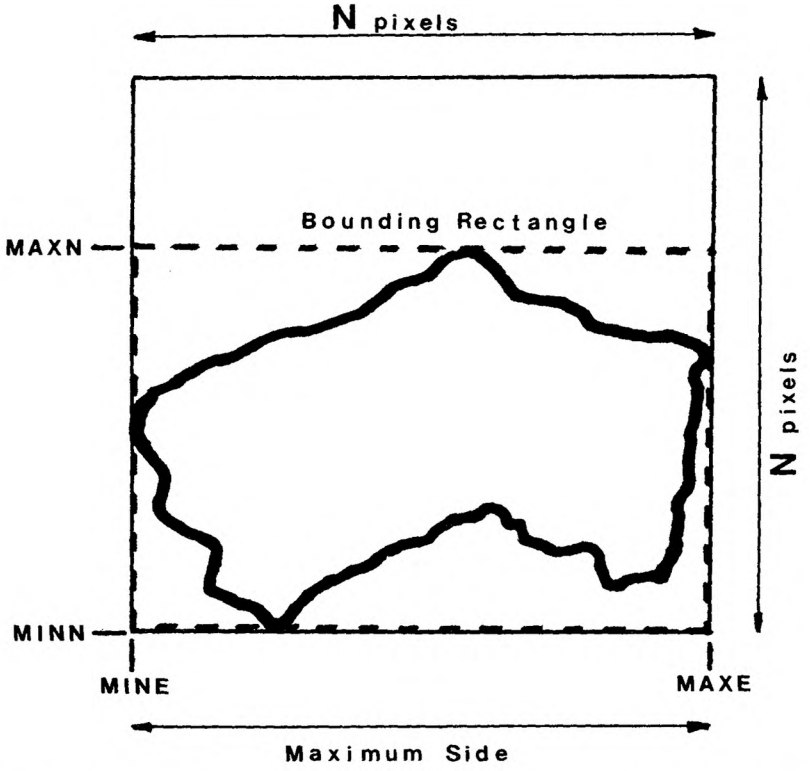


Fig 5.14 Area bounding rectangle inside NxN rasterized array.

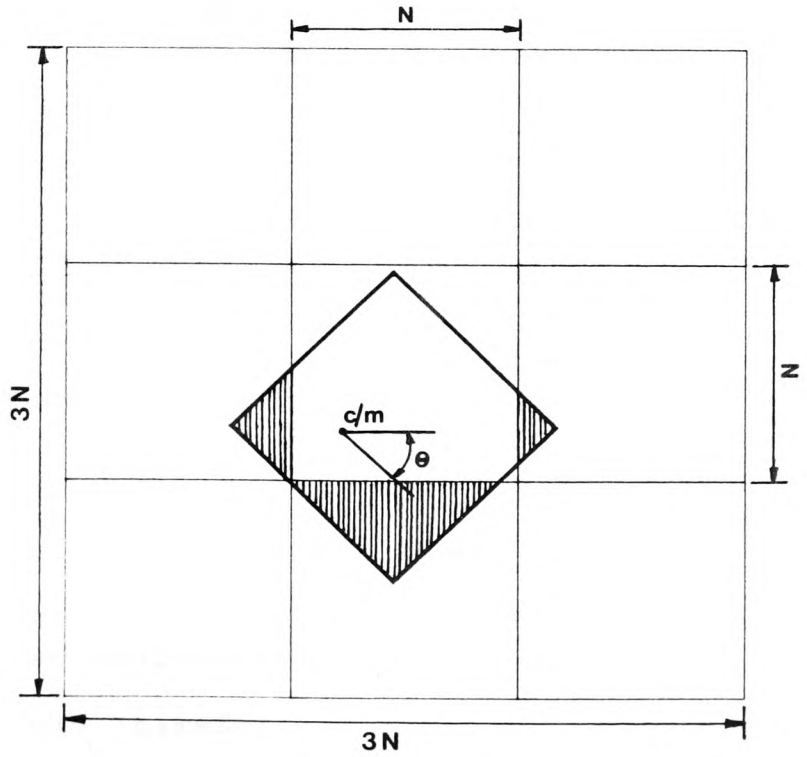


Fig 5.15 Rasterized (NxN array) area rotated about centroid so that the major axis is now lying horizontal inside a 3Nx3N array.array.

5.4.3 HORIZONTAL AREA LABEL PLACEMENT [IMPLEMENTED]

To prevent any part of a horizontal label being placed outside the bounding rectangle of the area (Section 5.4.2), the centre of the label must be placed inside an internal border region as depicted in Fig 5.16. The major axis of the area cuts this border region in two places and defines the limits of a reference line which is used in the computation of horizontal area name placement. There are six possible arrangements for the major axis crossing the border region (Fig 5.17), and the possible intersection points are:

```
LEFTx=L/2                          LEFTy=yo+(LEFTx-xo).Tan(Q)

RIGHTx=N-L/2                       RIGHTy=yo+(RIGHT-xo).Tan(Q)

UPx=xo+(UPy-yo)/Tan(Q)             UPy=N-W/2

DOWNx=xo+(DOWNy-yo)/Tan(Q)         DOWNy=W/2
```

where:

L = Label length (pixels). W = Label width (pixels).

xo = X centroid (pixels). yo = Y centroid (pixels).

Q = Angle of maximum moment of inertia.

N = No of pixels in original array.

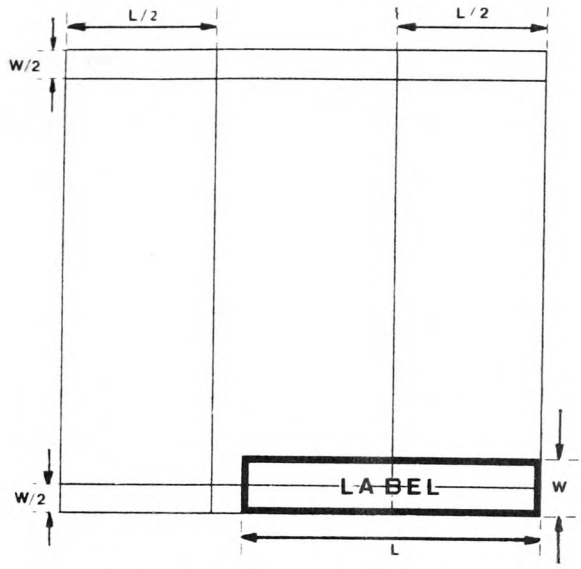


Fig 5.16 Label centre must lie within an internal border region within the NxN rasterized array.

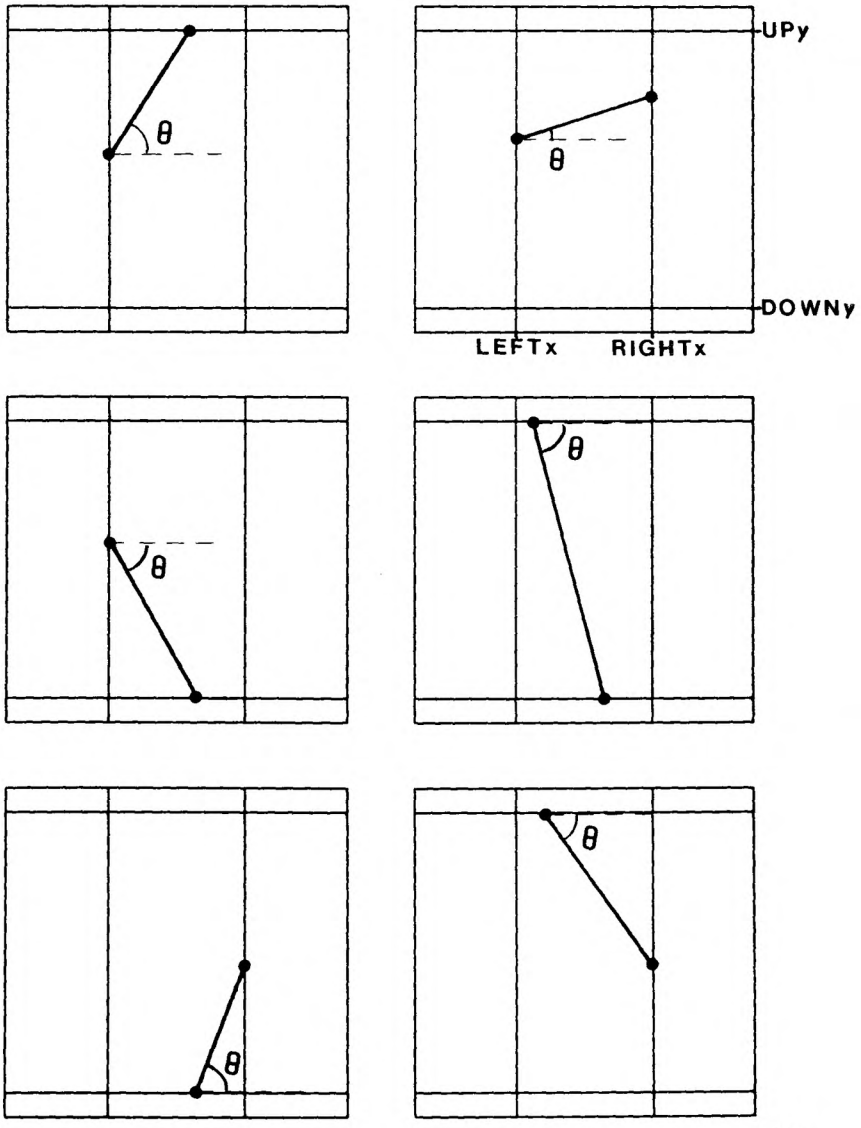


Fig 5.17 Intersection of area's major axis with internal border of NxN rasterized array.

Twenty equi-spaced lines perpendicular to the major axis are generated between the two points and the rectangular label is slid up and down each (Fig 5.18) attempting to find the best placement inside the area. Fig 5.19 shows how the goodness of fit value for a horizontally placed area label can be found. The value is the square root of the square of the sum of the differences of area pixels to the left and to the right of the label, and the square of the sum of the differences of area pixels above and below the label (Root sum of the squares).

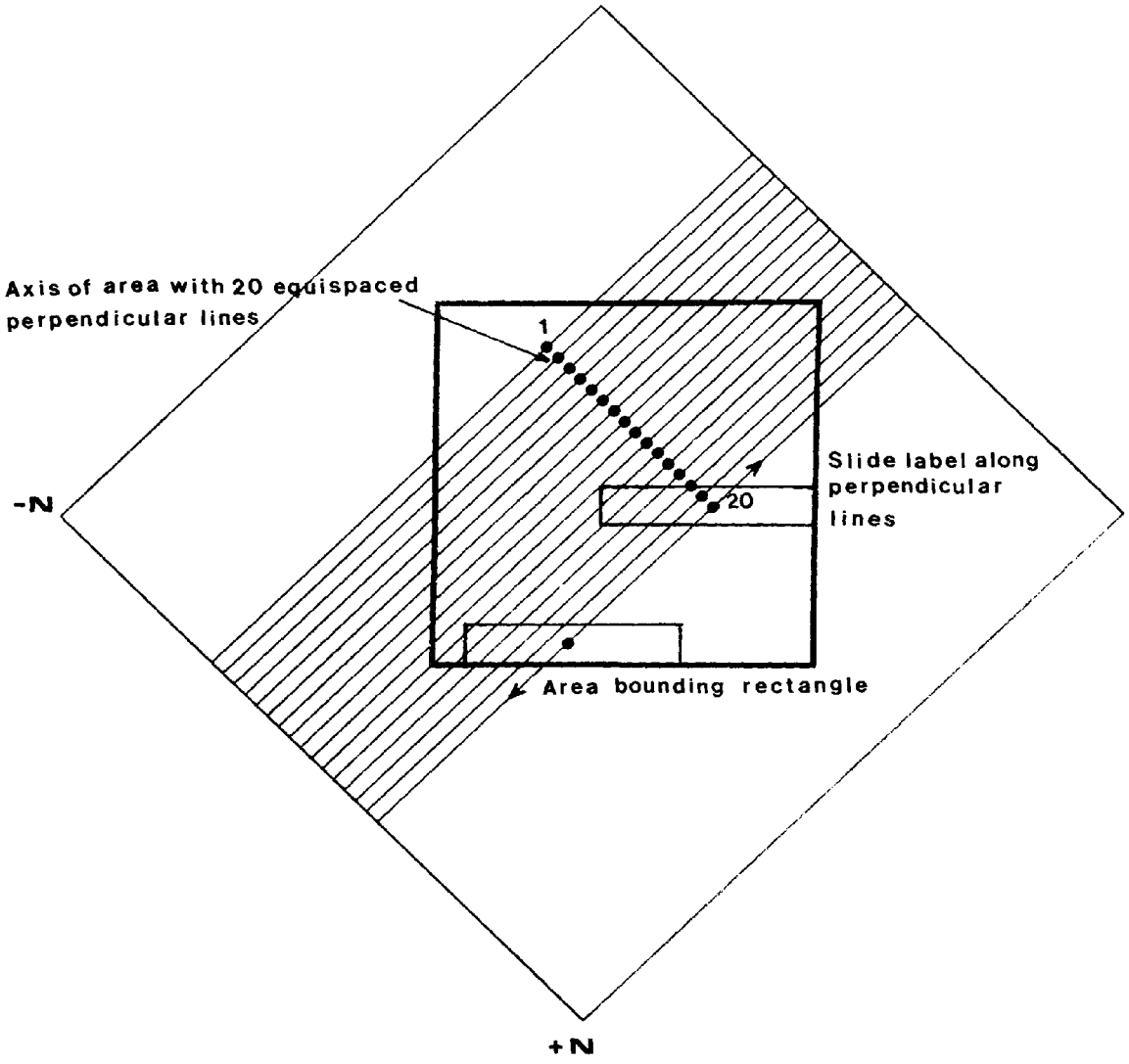
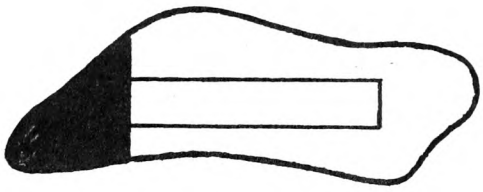


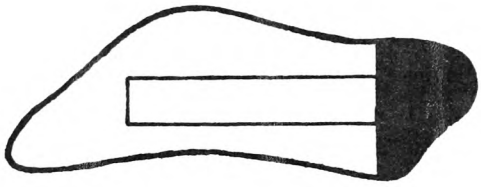
Fig 5.18 Slide label along 20 equispaced lines.

Pseudo code for algorithm to find and evaluate up to twenty area label positions:

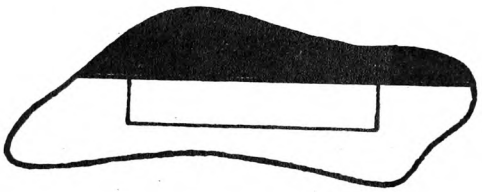
```
for 20 equi-spaced lines perpendicular to major axis
  initialise best_fit(perpendicular)
  initialise x_pos(perpendicular)
  initialise y_pos(perpendicular)
  for all positions +/- N pixels along the current
perpendicular
    compute_label_position(x,y)
    if current position inside label border region then
      if label totally inside area then
        evaluate goodness_of_fit
      endif
      if goodness_of_fit < best_fit(perpendicular) then
        best_fit(perpendicular):= goodness_of_fit
        x_pos(perpendicular):= x
        y_pos(perpendicular):= y
      endif
    endif
  nextfor
nextfor
```



Σ_{LEFT}



Σ_{RIGHT}



Σ_{UP}

Σ_{DOWN}

$$GOOD = \sqrt{(\Sigma_{LEFT} - \Sigma_{RIGHT})^2 + (\Sigma_{UP} - \Sigma_{DOWN})^2}$$

Fig 5.19 Definition of goodness of fit of a label inside an area.

The list of positions, contained in the arrays "x_pos" and "y_pos", are then sorted so that the most preferred has the smallest best fit value. The positions are then converted into database coordinates. A count is kept of the number of valid positions found inside the area. A position is valid as long as its best fit value has changed from its initial value. Like point and line placement algorithms (Sections 5.2 and 5.3), the positions found do not take into account underlying detail other than the feature concerned, so several of the positions found may be invalid.

5.4.4 DIAGONAL AREA LABEL PLACEMENT [IMPLEMENTED]

This algorithm is similar to the previous one, except that it uses the 3N x 3N array and the label is parallel to the major axis of the area which lies horizontal in the rotated array. Once the label positions are found, these must be rotated back into the N x N array and then converted into the database coordinates.

5.4.5 CURVED AREA LABEL PLACEMENT [DEMONSTRATED]

It was decided to use a technique similar to Hirsch and Glick's (1983) method of curved area placement. A least squares curve fitting algorithm is used to fit

either a parabolic or cubic curve through the mid-points of the strips making up the area. The letters constituting the label are aligned along the direction of the curve.

5.4.6 SECTION DISCUSSION

This section of the chapter has discussed methods of placing names inside areas. The first two algorithms find a range of positions for horizontally and diagonally placed labels whereas the curved label algorithm only finds one location for each curve (parabolic and cubic) it fits through the area. To evaluate the acceptability of the area label placement technique some example placements, similar to those in Figs 5.20 to 5.22, were sent down to the Ordnance Survey headquarters. It was commented that many of the area name placements approached closely to the ideal locations as interpreted by the Ordnance Survey (King, 1987). The most important fact which emerged from this exercise was the importance of choosing the correct label configuration, type size and style for a particular area. Fig 5.20-22 illustrate some examples of area label placement generated using the three algorithms described in this section of the chapter.

A technique of placing area labels outside or partially outside the area was not investigated. One possible means of achieving this though is to apply point

name placement to areas, with the point lying at the centroid of the area and the radius of proximity varying with boundary distance from the centroid and a fixed offset. This would enable a discrete number of positions to be found along the boundary and the presence of the area in a raster image of the map could be used for deciding how much to offset the label from the boundary by.

The three area placement algorithms investigated by the author produce satisfactory results as long as the correct label configuration is used for a particular area shape and size. The elongation and angle of the area major axis could be used to distinguish which areas are suitable for horizontal or diagonal configuration labels (Section 8.4). The use of curved configuration labels is more difficult to determine and may require a study of the "wiggliness" of the mid-points of area strips (See section 5.4.5).

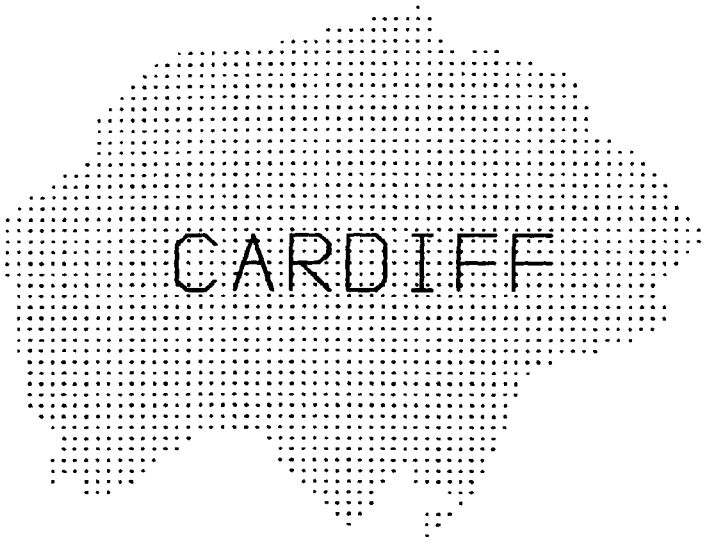


Fig 5.20 Horizontal area name placement.

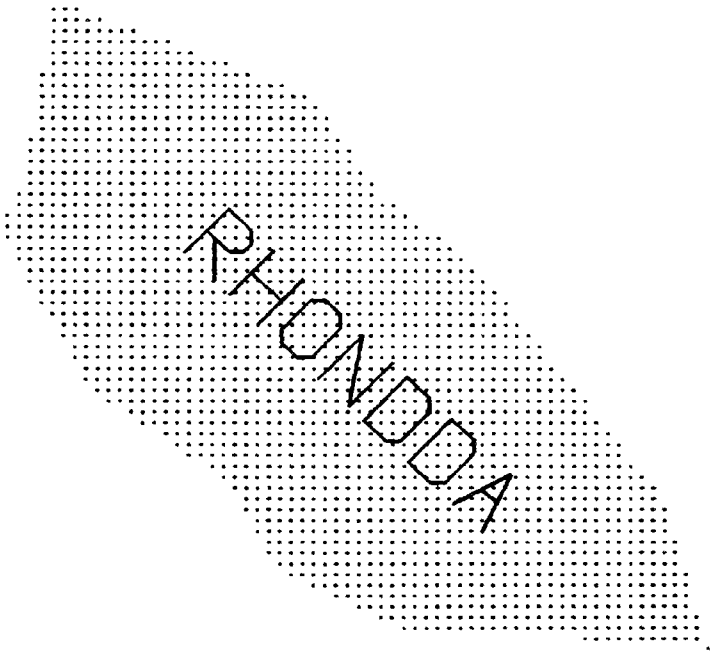


Fig 5.21 Diagonal area name placement.

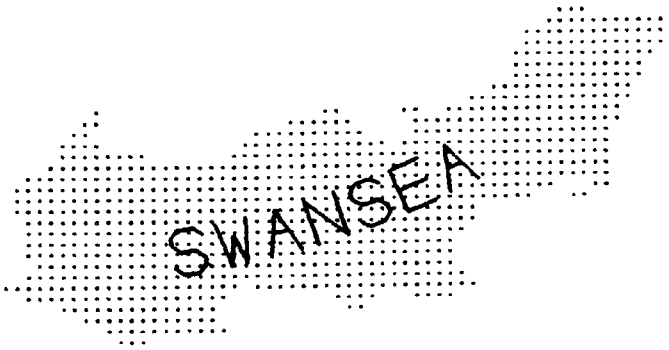


Fig 5.22 Curved area name placement.

5.5 CHAPTER CONCLUSION

This chapter has identified a suitable set of label configurations for a versatile automated name placement system to use on a wide range of maps. It has described how to quantise most of these into 20 placement positions no matter what type of feature the label represents.

To demonstrate automated name placement it is only necessary to implement some of the label configurations described in this chapter. These include horizontal point labels, diagonal and horizontal line labels, which are improved versions of existing automated name placement techniques. Additionally the new algorithm for horizontal and diagonal area label placement, devised by the author, will also be implemented along with the use of multiple lined labels. A combination of these implementations are used in conjunction with the two name placement systems to be described in chapters 6 and 7.

CHAPTER 6

"LABPOS" - A PROTOTYPE NAME PLACEMENT SYSTEM

6.1 INTRODUCTION

6.1.1 AIMS

In order to gain familiarity with some of the existing techniques in automated name placement, and to evaluate new techniques, a prototype name placement program, "LABPOS" was developed during the initial year of the research. The scope of this name placement program was limited to one type of map, the 1:625000 Route Planner map, the data for which was kindly provided by the Ordnance Survey and transferred onto the VAX 11/785 at the Polytechnic of Wales. LABPOS was designed to work directly from the Ordnance Survey or O.S. database with no intermediate data processing stages.

However a few restrictions were applied to the classes of feature that could be labelled using the prototype name placement system. For instance name placement was restricted to "roads and settlements" because a method of placing area names and curved names had not been developed at this early stage in the research. Also because the Ordnance Survey database was

divided into three datasets: "roads and settlements", "contours" and "administrative areas", to save time accessing the database, only the "roads and settlement" dataset was accessed. Unfortunately coastline and river data were stored in the "administrative area" dataset and so these features were not included in the raster data. However for the purposes of clarity, the coastline has been included on the map examples displayed in this chapter, but the name placement process does not regard the coastline as being present.

The existing name placement techniques utilised include:

1) Constraints that labels are placed at predefined positions with respect to their features (Section 4.3).

2) Ranks or priorities assigned to label positions to define the preferred order of placement (Fig 6.6 and 6.7).

3) The use of raster data to detect label overlaps with underlying detail (Section 4.4.2).

4) Label splitting (Pfefferkorn et al, 1985).

The new aspects to be investigated include:

1) User definable name placement rules.

2) Optimization of the name placement program.

3) The storage of feature classified raster data in bit planes (Section 3.3).

4) A new iterative placement algorithm.

The main advantage of LABPOS over many existing name placement systems is the ease with which user controllable rules and parameters can be altered. These in turn affect the map design and efficiency of the name placement. The rules can be tailored to meet the cartographic requirements of the map designer within the constraints provided. This is performed through a process of optimization (Section 6.9.2) which is not described in any previously published accounts of automated name placement systems.

The use of raster bit planes provides a means of distinguishing the classes of features obscured by labels covered at different positions (Section 3.3.2). It also allows the capability to construct different kinds of maps from a selection of user specified raster planes (Freeman,

1985), however this will not be used by the author. The importance or priority of each feature class can be assigned to the respective raster bit plane and easily altered if desired.

A new iterative technique of name placement was investigated because when LABPOS was being developed the only other known iterative name placement technique, by Hirsch (1982), had several shortcomings. According to Basoglu (1984), Hirsch's technique required many iterations to work and from the published results it appears that label ambiguity occurred despite the use of fixed size buffer zones.

6.1.2. OVERVIEW OF LABPOS

The LABPOS program can be broken down into the basic programming modules represented in the flow chart shown in Fig 6.1. This chapter though will be primarily concerned with the user definable name placement rules, the conversion of the vector map data into a raster image and the name placement strategy.

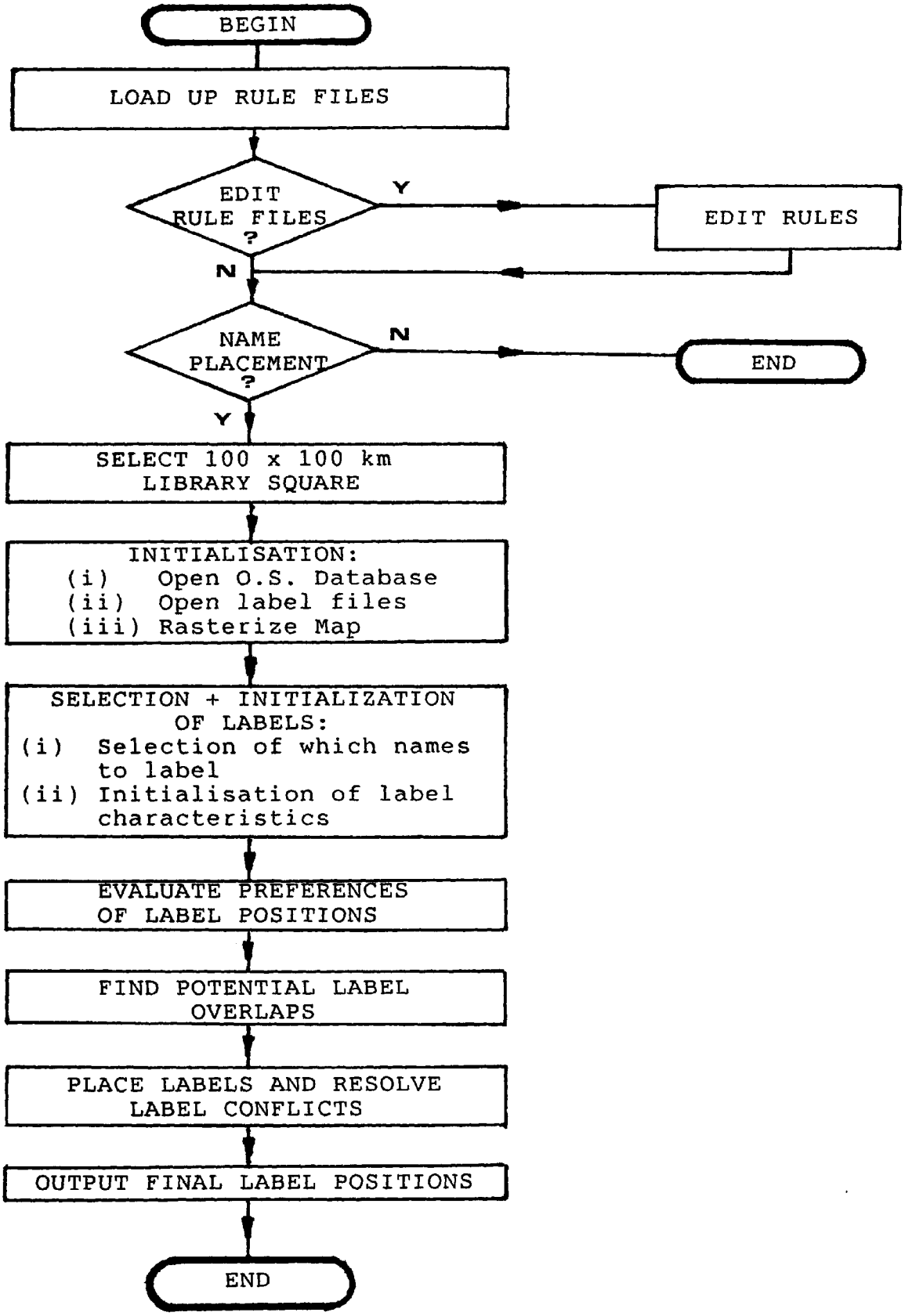


Fig 6.1 LABPOS modular flowchart.

During the selection of which names to label, some multiply-worded names may prove to be too long and are likely to be difficult to place. Such labels are identified and flagged as labels to be split according to criteria to be discussed in section 6.4.4.3.

Having selected which names to use as labels, all available label positions are examined and their suitability with respect to underlying map features determined. This is done by counting pixels contained within label rectangles (Section 6.5). LABPOS treats all labels as rectangles which encompass the letters of the label concerned (Fig 6.2). Some of the possible label positions can be eliminated because they obscure too much underlying detail. The remainder are sorted into an ordered list of decreasing preference and stored for each label.

Before name placement can take place, the program determines which labels may potentially overlap each other. This is achieved by placing a bounding rectangle around the range of all possible positions for each label and then testing to see which of these rectangles overlap (Section 6.6.2).

Unlike all previous methods, labels are not restricted to placement in an order of point labels prior to line labels. Instead LABPOS places all labels in their

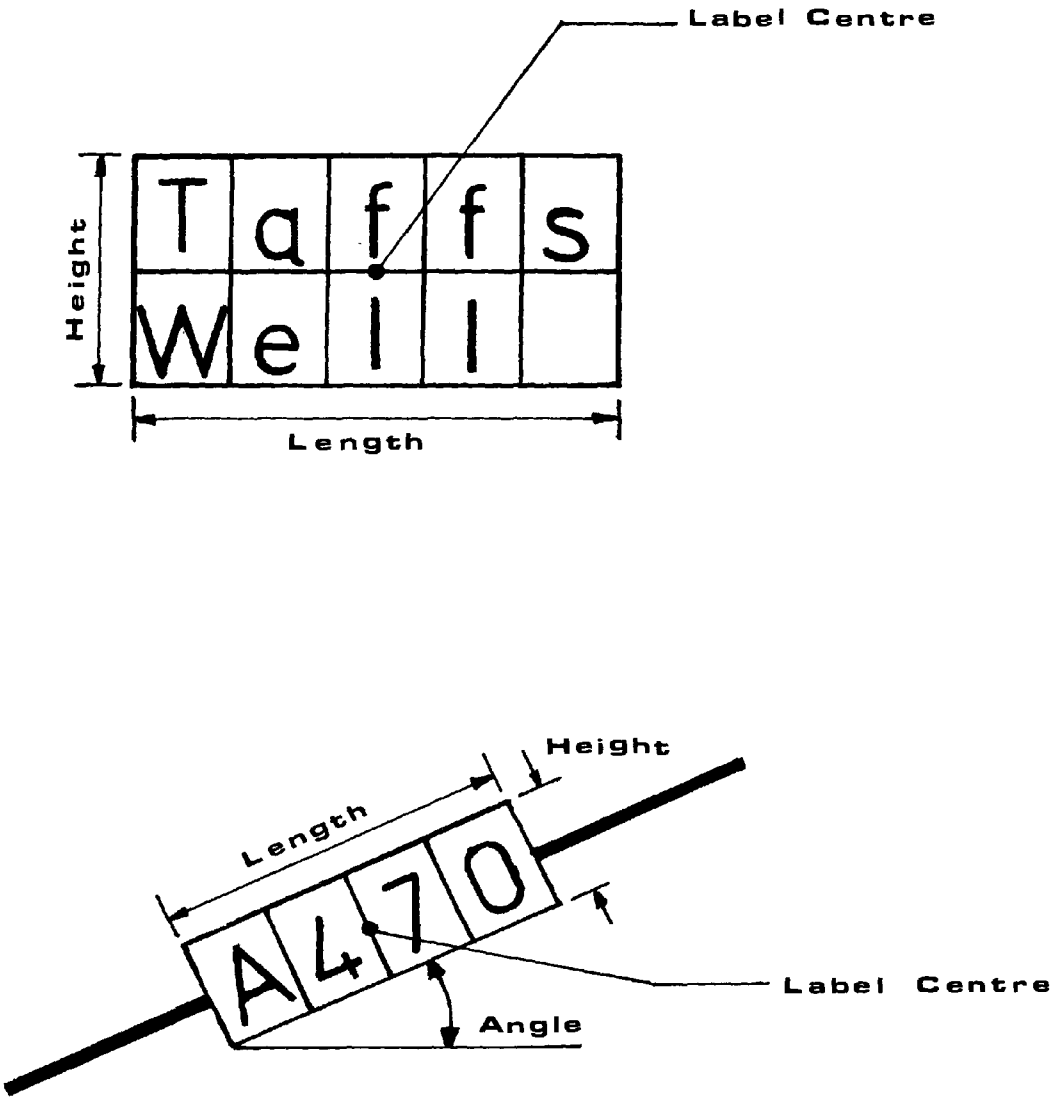


Fig 6.2 LABPOS representation of horizontal point and diagonal line labels.

most preferred positions and then concentrates on resolving label:label conflicts through a series of iterations. To avoid endless loop situations which can occur during iterations, the more that a label is found to be in conflict, the less likely its neighbouring labels are to be placed in overlap with it during subsequent iterations (Section 6.7).

To reduce ambiguity, emphasis is placed on moving labels as far apart as possible but relaxing this constraint in conflict situations which are difficult to resolve. The process of iteratively placing each label continues until the label has been satisfactorily placed or until there is no point in continuing due to an unresolved conflict situation, defined by a user specified number of placement attempts. If the latter occurs the label is fixed in its most favourable position with respect to underlying detail and its neighbouring labels are forced to move out of the way instead.

Once all labels have been placed, their positions are output in the form of a sequential ASCII file which is easily portable (Section 6.8). Any remaining labels which still remain in overlap or are too close to an adjacent label are flagged for future reference.

6.2 USER DEFINABLE NAME PLACEMENT RULES

User definable name placement rules take the form of specifications and rules which govern name placement. The rules can be edited through menu options and the revised rules saved.

The rules are divided into two types, high level rules and low level rules. The high level rules form the majority and are of direct consequence to name placement, the low level rules are really specifications more concerned with the efficiency of the system rather than name placement. The distinction between these two levels is not clear cut since the high level rules can affect program efficiency and some of the low level rules can affect the aesthetics of name placement. A selection of high level rules is given below:

Rule **[6.2]** can be used to indicate to LABPOS how important it is to avoid placing labels over certain classes of feature.

Rule **[6.4]** lets the user select which features to label according to feature code.

Rule **[6.5]** is used to select which lines to label according to criteria dependent upon line length and label size.

Rule **[6.7]** specifies how far point labels lie from their symbols.

Rule **[6.8]** can be used to encourage label splitting.

Rule **[6.9]** is used to specify the justification of label splitting with respect to the point symbol.

Rule **[6.10]** governs the preferred order of placement around a point for point labels.

Rule **[6.11]** governs the preferred order of placement along a line for line labels.

Rule **[6.12]** lets the user specify what proportion of underlying map detail to label area is permitted before a label position cannot be used.

Rule **[6.13]** allows the user to specify ideal minimum separation distances between placed labels.

There are also some low level rules, for instance the user can define the raster feature widths and bit planes [6.1], and raster resolution [6.3]. The latter affects the run time of the program.

Fifteen specifications and rules are available to the user and are stored in the form of parameters in two files, "RULE" and "FEAT_RP". The "RULE" file contains general specifications and rules (Table 6.1) and the "FEAT_RP" file contains specifications dependent upon feature code (Table 6.2). All these rules will be covered in more detail later on in the chapter.

Table 6.1 "RULE" file record contents

Record No.	Name	Rule No.	Description
1	NN	6.3	Size of raster grid.
2	PRIOR(1)	6.2	Priority of bit plane 1: "B roads".
3	PRIOR(2)	6.2	Priority of bit plane 2: "A roads".
4	PRIOR(3)	6.2	Priority of bit plane 3: "Main routes".
5	PRIOR(4)	6.2	Priority of bit plane 4: "Motorway".
6	PRIOR(5)	6.2	Priority of bit plane 5: "Settlements".
7	OUT	6.2	Over edge priority.
8	FILTER	6.5	Line label filter flag: T=filter on, F=filter off.
9	BUFFER	6.5	Line label buffer zone in character block widths.
10	POINT_ORDER(1)	6.10	Point label position 1.
..	.....	...	.....
..	.....	...	.....
25	POINT_ORDER(16)	6.10	Point label position 16.
26	LINE_ORDER(1)	6.11	Line label position 1.
..	.....	...	.....
..	.....	...	.....
41	LINE_ORDER(16)	6.11	Line label position 16.
42	BUFFER	6.13	Buffer zone units T=character widths, F=metres.
43	BUFFER_X	6.13	Vertical sides buffer zone.
44	BUFFER_Y	6.13	Horizontal sides buffer zone.
45	PROX_LETTER	6.7	Radius of proximity: T=character widths, F=metres.
46	PROX	6.7	Radius of proximity.
47	IPIX	6.12	Dense space threshold (%).
48	PERCENT	6.8	Threshold needed to split a name (%).
49	NOSPLIT	6.8	Label splitting flag: T=not split, F=split.
50	JUSTIFIED	6.9	Left, centre or automatic justification (1,2 or 3).
51	MID_LIMIT	6.14	No. of placement attempts before weighting.
52	LIMIT	6.15	No. of placement attempts before giving up.

Table 6.2 "FEAT_RP" record structure

Field name	Rule No.	Description
FC	-	Feature code.
FT_NAME	-	Feature code name.
FT_WIDTH	6.1	Feature width (m).
FT_BIT	6.1	Feature bit plane.
LET_HT	6.4 & 6.6	Letter height (m).
LET_WT	6.6	Letter width (m).
LOW_CS	6.6	Lower case letter height (m).
LETTR_DS	6.6	Letter descender height (m).
BLOCK_HT	6.6	Letter block height (m).
BLOCK_WT	6.6	Letter block width (m).

6.3 RASTERIZATION

6.3.1 DEFINITION OF FEATURE WIDTHS AND BIT PLANES - RULE [6.1]

A 100 x 100 km square area of the Route Planner map is initially rasterized into a square two dimensional array (four byte integer) of sides N pixels, each pixel being of width 100/N km.

During rasterization, map features can be represented at their appropriate scaled line gauge, or if necessary, their line thickness may be increased to have greater influence on the name placement process. For example, if we wish to avoid a settlement label being placed too close to another similar feature, then the width should be increased such that the symbol's radius in the raster image is slightly greater than the label's radius of proximity (Note that this is not allowed to affect the labelling of the labels own feature - see section 6.5.3).

When accessing this rule through the LABPOS rule editing menu, two other aspects related to the code of the features can be adjusted. Firstly, the user can allow for features of different codes to be placed into different bit planes, however this would only be necessary if additional feature codes became available or amendments are made to the definition of existing features codes

present in the Ordnance Survey Route planner database. Secondly, if a feature of a particular code is not to be rasterized, it is flagged by giving it a width of zero.

6.3.2 FEATURE CLASS PRIORITY - RULE [6.2]

When label positions are investigated, pixel counting is used to find out what proportion of underlying features are present. An allowance must be made for feature importance, for instance labels are less frequently placed over A class roads than over B class roads. To cater for feature importance, instead of counting pixels occupied by a feature as single units, the priorities for features present in each pixel are summed. Thus if we want a certain class of feature to stay fairly free from overlapping labels, then we give the priority for that class a high number, for example 2.0 which means that a pixel counts twice as much as an ordinary pixel. A priority number as high as 999 will, if found at a label position, make that position appear to be too full of underlying feature detail and therefore not suitable.

To avoid having to quote priorities for all types of features on the Ordnance Survey Route Planner map, features have been reclassified into just six classes which are then allocated separate bit planes in the raster data. The first five feature classes are "B class roads",

"A class roads", "main routes", "motorways", and "settlements". A sixth class, "beyond the map border", is not strictly a map feature, but is included in order to avoid placing names across a map edge.

6.3.3 RASTER RESOLUTION - RULE [6.3]

One way of speeding up the name placement program is to reduce the raster grid resolution. This enables both the vector to raster and pixel counting processes to be performed more quickly because there are less pixels to handle. However a reduced raster resolution means that both cartographic features and labels are not being as accurately represented as at higher resolutions and this results in fewer possible label positions being found and hence the likelihood of more unresolvable conflicts. Section 6.9.2.2 will discuss the choice of suitable raster grid square sizes.

6.4 SELECTION AND INITIALISATION OF LABELS

6.4.1 INTRODUCTION

Although the Ordnance Survey database provides a definitive list of all named features, it is necessary to define criteria for selecting which names should be labelled. Once selected, a label is initialised by determining its configuration, dimensions and assigning it a record in the "LABEL1" file which is used to store label information (Table 6.3).

6.4.2 SELECTION BY FEATURE CODE - RULE [6.4]

One means of selecting names is based upon the class of the named feature. To prevent features of a particular class from being labelled, a letter height of zero is inserted into the relevant record in the "FEAT_RP" file (Table 6.2).

6.4.3 LINE LABEL SELECTION - RULE [6.5]

When using LABPOS to label the Route Planner map, the only named linear features of any significance are roads. The placement configuration of road labels is hard encoded such that Motorways and A class roads have their labels

Table 6.3 "LABEL1" record structure:

Field name	Description
LAB	Label serial No. in the "LABEL1" file.
NID	Name Identification No. from the Ordnance Survey Route Planner database.
FSN	Feature serial number in the Ordnance Survey Database.
FCODE	Feature Code.
L	Length of label rectangle (10m units).
H	Height of label rectangle (10m units).
RD	Radius of proximity for point labels (10m units).
ANG	Orientation of label measured anti-clockwise from East (degrees).
TYPE	0 if label refers to a line and 2 if it refers to a point feature.
SPLIT	0 if label not split, 1 if split.
NAME	A left justified character string containing the name of the label, terminated by a "*".
EAST	Eastings of centre of label (10m units).
NORTH	Northings of centre of label (10m units).
POS	An array of 16 preferred positions No's for the label.
OVRLP	An array of up to 32 record No's of labels which could possibly overlap with this label.
NO	A count of how many label record No's are in the OVRLP field.
FL	Overlap flag: 0=no overlap, 1=possible overlap.

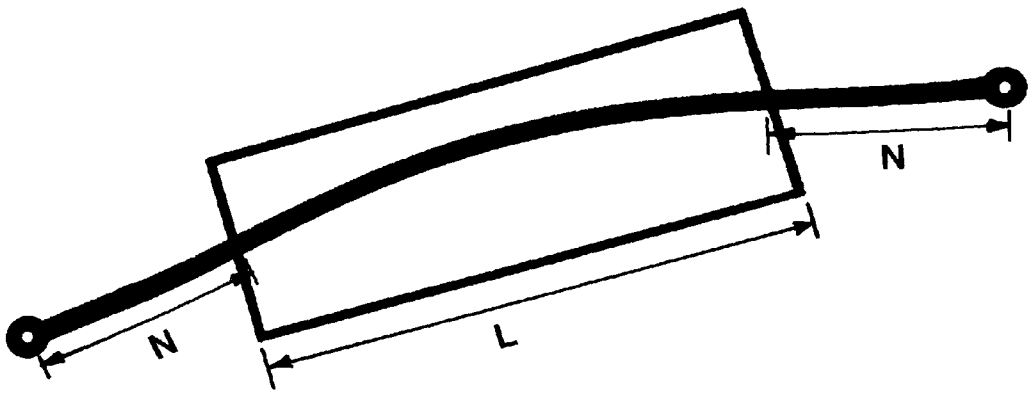


Fig 6.3 Minimum length of line suitable for labelling. The line must be longer than the label length L plus twice the buffer distance at each end (N character widths).

placed parallel and centrally on the road and B class road labels are placed parallel but offset.

The user has the option of either letting LABPOS place all named line features or to selectively filter those which are below a certain length. The former can result in the map becoming very crowded with labels unless the line names have been pre-filtered beforehand in the Ordnance Survey Route Planner database. The latter produces more reasonable results providing that the minimum line length threshold has been selected carefully.

The minimum line length threshold (Line buffer) can be specified by the user in terms of a buffer distance which must be kept clear at each end of every line label (Fig 6.3). This restricts the selection of lines to those at least as long as the label itself and by increasing the buffer distance, this has the effect of reducing the number of links which can be labelled.

6.4.4 LABEL INITIALIZATION

6.4.4.1 DEFINITION OF TEXT PARAMETERS - RULE [6.6]

The user can specify letter heights and widths and other parameters describing the letter characteristics used in the graphical representation of the letters such

as lower case letter height and letter descender height. The most important parameter needed by LABPOS though, with respect to letter characteristics is known as letter block height and block width. Block width is the distance between the start of one letter and the start of a next (In LABPOS this includes the space between letters). The block height is the distance between the bottom of one line of text and the bottom of the line below (Fig 6.4). When plotting out labels after placement, the letters should be placed as centrally as possible inside their block boundaries allowing for any offsets due to letter descender height.

On some occasions, labels can be enlarged from their standard feature class size to reflect a feature's importance. The Ordnance Survey database dictates when this should occur and an allowance is made for this in LABPOS.

6.4.4.2 LABEL RADIUS OF PROXIMITY FOR POINT LABELS - RULE [6.7]

In a very crowded map, it might be advisable to make the radius of proximity small so as to allow more free space and emphasize the relation between labels and their associated points. The user can specify the radius of proximity either in terms of metres or character block

widths.

6.4.4.3 LABEL SPLITTING CRITERIA - RULE [6.8]

Some labels consisting of multiple words have an improved chance of being placed if they are split into two lines, thus shortening label length but doubling the height. If a label is potentially "splittable" according to rules [2.77], [2.79], [2.80] and [2.81], then the sum of bit plane pixel priorities in the permissible placement region (Fig 6.5) is computed for split and non-split configurations. The user can specify whether label splitting is allowed and also the percentage improvement threshold needed for split over non-split configurations, in underlying free space, before a label is actually split.

6.4.4.4 SPLIT LABEL JUSTIFICATION - RULE [6.9]

When label splitting is permitted, the user can specify whether the label is to be left justified, centred or automatically justified according to which side of the point the label lies on.

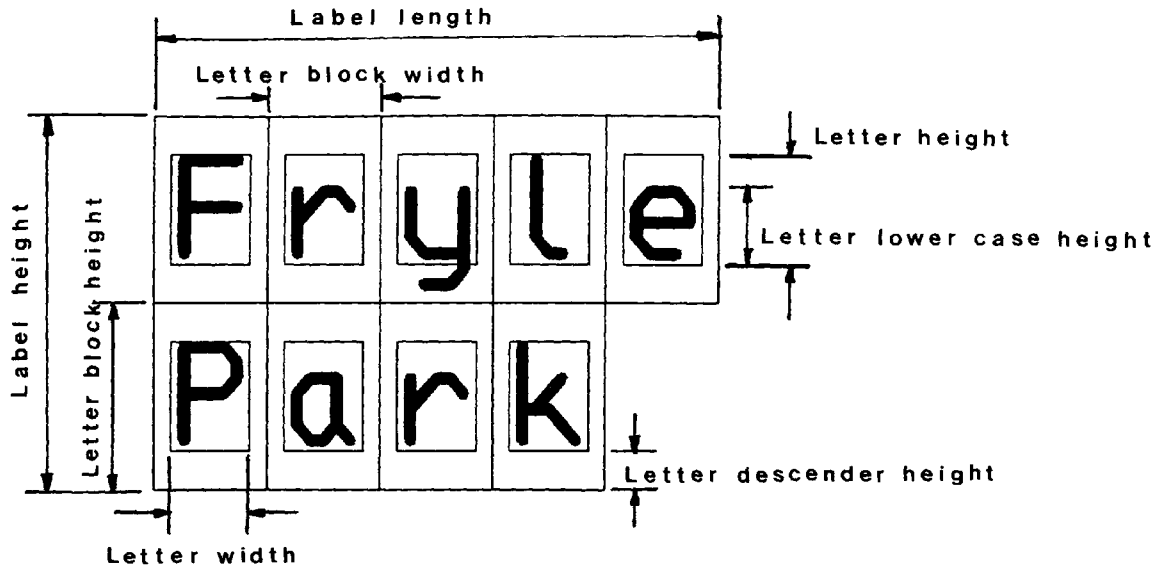
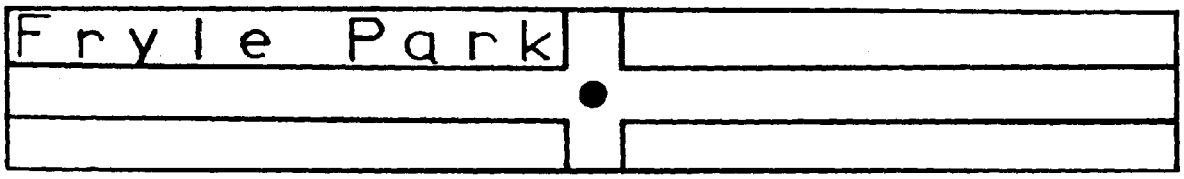
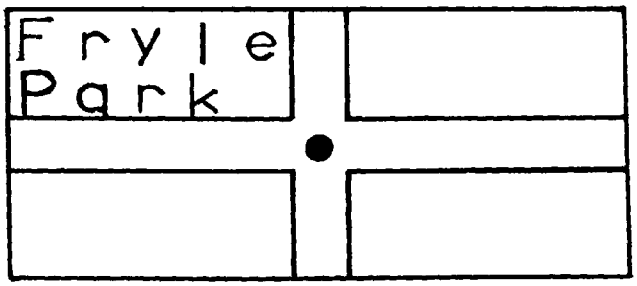


Fig 6.4 Block letter representation of labels.



NON-SPLIT



SPLIT

Fig 6.5 Region of containment for a point label in non-split and split configurations.

6.5 LABEL POSITION PREFERENCE

6.5.1 INTRODUCTION

Following the completion of label selection and initialisation, the next stage is to evaluate the preference of label placement positions prior to placement. There are two types of placements, firstly those which lie in totally "free space" on the map and secondly those where the label obscures some underlying detail or "dense space". Once the preferences for all positions for a label have been computed, with respect to underlying detail, these are sorted and the position numbers are written to the "LABEL1" record. When labels are placed, they are positioned in the order defined by the "LABEL1" records. Positions lying in free space are preferred to those lying in dense space.

6.5.2 PREFERRED ORDER OF PLACEMENT IN FREE SPACE

6.5.2.1 ORDER OF PLACEMENT OF A LABEL AROUND A POINT - RULE [6.10]

Because cartographers may differ in opinion as to the exact order of placement preference for point labels in free space, it is necessary to be able to define this order. This high level rule allows a user to define the

preference of up to 16 positions around a point (Fig 6.6). If the user wishes to prevent any position being used, then the position may be left out of the list and another more preferred position repeated.

6.5.2.2 ORDER OF PLACEMENT OF A LABEL ALONG A LINE - RULE [6.11]

As with rule [6.10], label positions along a line vary in preference. Although lines can be thought of as symmetrical features with respect to name placement, each end being treated equally, it is possible to favour labels being placed towards the centre or ends of the line (Fig 6.7). A class road labels are placed parallel and centrally on the line, B class road labels are placed parallel but alternately offset above and below the line. If the user wishes to prevent certain positions being used, then the same method adopted in rule [6.10] is applied.

6.5.3 ORDER OF PLACEMENT IN OCCUPIED MAP SPACE - RULE [6.12]

When LABPOS selects a placement position, some positions will be unsuitable due to the presence of either very important map features such as towns or a high

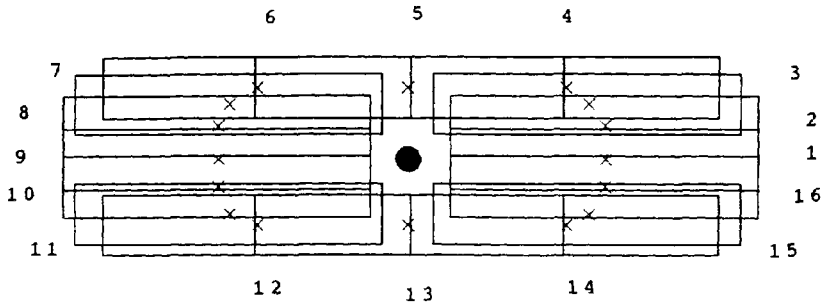


Fig 6.6 LABPOS point label positions (Positions refer to the centre of label, marked by a cross).

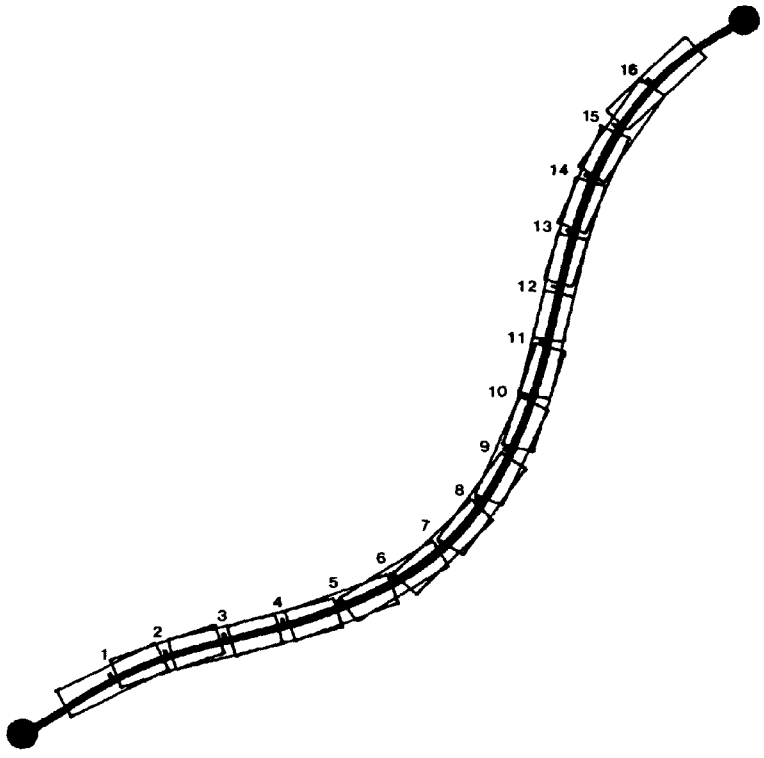


Fig 6.7 LABPOS line label positions (Positions refer to the centre of label).

feature density area of the map. Label placements are evaluated by summing the pixel priority values belonging to underlying map features within a label rectangle and dividing by the total pixel area of the rectangle.

Because pixel counts may include pixels belonging to the feature which is being labelled, this could create a false impression of the unsuitability of the position. This is particularly relevant to A class roads where labels must lie in the middle of the road. To avoid this false impression, a raster bit plane is set aside for use in flagging or masking all pixels pertaining to that feature so that they are not counted (Fig 6.8). However, to avoid the accumulation of masks, these must be erased prior to their use on the next feature. This is achieved by keeping a record of a bounding rectangle around each mask and by initialising the previous mask bit plane contents to zero.

Another essential requirement is that positions are not used where the label lies over important features such as towns. Because high priority values can be assigned to a raster bit plane that contains such features, as soon as the count of pixel bit plane priorities exceeds that value, the position for that label can be flagged as illegal. The user can specify the "dense space threshold" or proportion of those pixels contained within that rectangular area which are allowed before the label

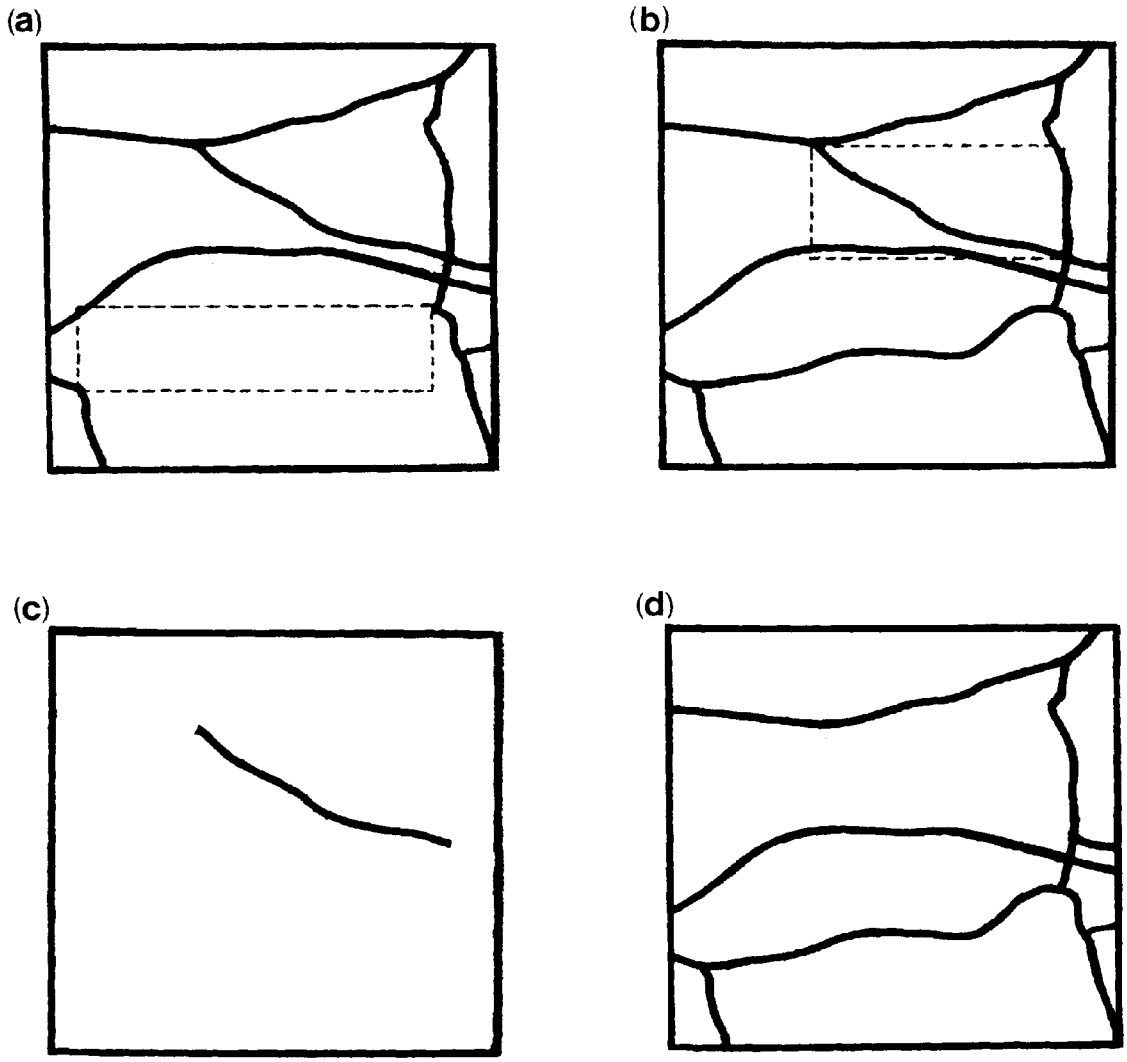


Fig 6.8 The process of feature masking:

a) Remove previous mask contained in current bounding rectangle.
b) Select new feature and compute bounding rectangle.
c) Mask out new feature in "rub-out" bit plane.
d) Use map with feature "masked-out".

position is deemed to be unsuitable for use. On the odd occasion, all positions may be flagged as illegal in which case the most preferred position with respect to underlying detail is selected.

6.5.4 PSEUDO CODE FOR EVALUATION OF LABEL POSITIONS

```
for all labels
  mask out current feature
  for all label positions
    find limits of label rectangle
    initialise pixel_count to zero
    initialise label_area to zero
    (* max_limit is the pixel priority of
       a settlement pixel *)
    while pixel_count < max_limit and pixels still to
                        count in label rectangle
      increment label_area by one
      if pixel not masked
        for all bit planes
          if bit plane contains feature then
            pixel_count:= pixel_count + bit plane priority
          endif
        next
      endif
    endwhile
    if pixel_count >= max_limit then
      (* flag label position *)
      weight:= 999
    else
      weight:= pixel_count / label_area
    endif
    store weight for current position
  nextfor
  sort positions by weight and ban any which are flagged
  update current label record with most preferred position
nextfor
```

6.6 LABEL OVERLAP DETECTION

6.6.1 INTRODUCTION

This section of the chapter discusses how overlaps between labels are detected in LABPOS. To alleviate the need to test repeatedly a label against all other labels on the map during name placement, before name placement begins a list of potentially overlapping labels is computed for each label. Once these lists of potentially overlapping labels are known, the real task of testing a label for conflict with another can begin.

The term "label conflict" refers to when the buffer or separation distance between labels is encroached. The term "label overlap" will also be used, this refers to when one label intersects another's bounding rectangle.

6.6.2 PRELIMINARY OVERLAP DETECTION

Initially bounding rectangles are computed which encompass all possible label position extremities for each label. For point labels, the bounding rectangles are enlarged to include a specified separation distance between labels, as will be discussed further in section 6.6.4 (Fig 6.9a). For line labels, the bounding rectangle can be defined by the minimum and maximum eastings and

northings of the line feature (Fig 6.9b). Although much of the space inside the bounding rectangle does not form part of the line label's permissible range of positions, the method of determining the bounding rectangle has the advantage of being both simple and quick. Next, each bounding rectangle is tested against each other bounding rectangle (Fig 6.10) and if they overlap then this fact is included in the list of potential overlaps for each label and written to the appropriate "LABEL1" record.

6.6.3 OVERLAP DETECTION

Overlap detection entails testing label A at position B with label C at position D to see whether they overlap each other at all. The algorithm used to detect label:label overlaps requires the coordinates of the four corners of each label, their bases, heights and tilt angles. A single logical parameter is used to pass back the information on whether the two labels overlap or not.

The algorithm starts by selecting the label with the biggest area as a clipping window and translates the corner coordinates of both labels so that the bottom left hand corner of the biggest label lies at the origin. If the biggest label is inclined at an angle then the corner coordinates of both labels are rotated through the angle but with an opposite sign about the origin. The biggest

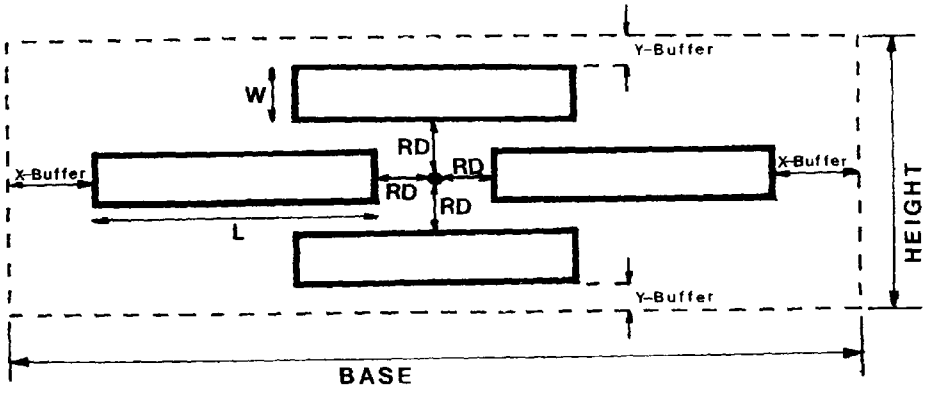


Fig 6.9a Region of potential overlap bounding rectangle for a point label.

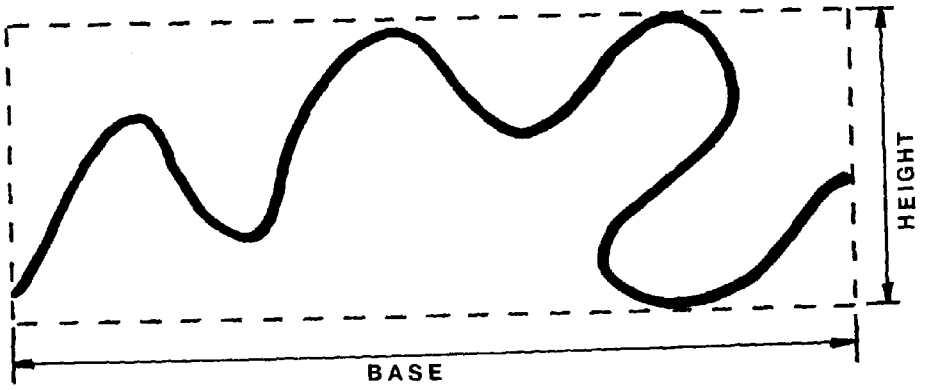


Fig 6.9b Region of potential overlap bounding rectangle for a line label.

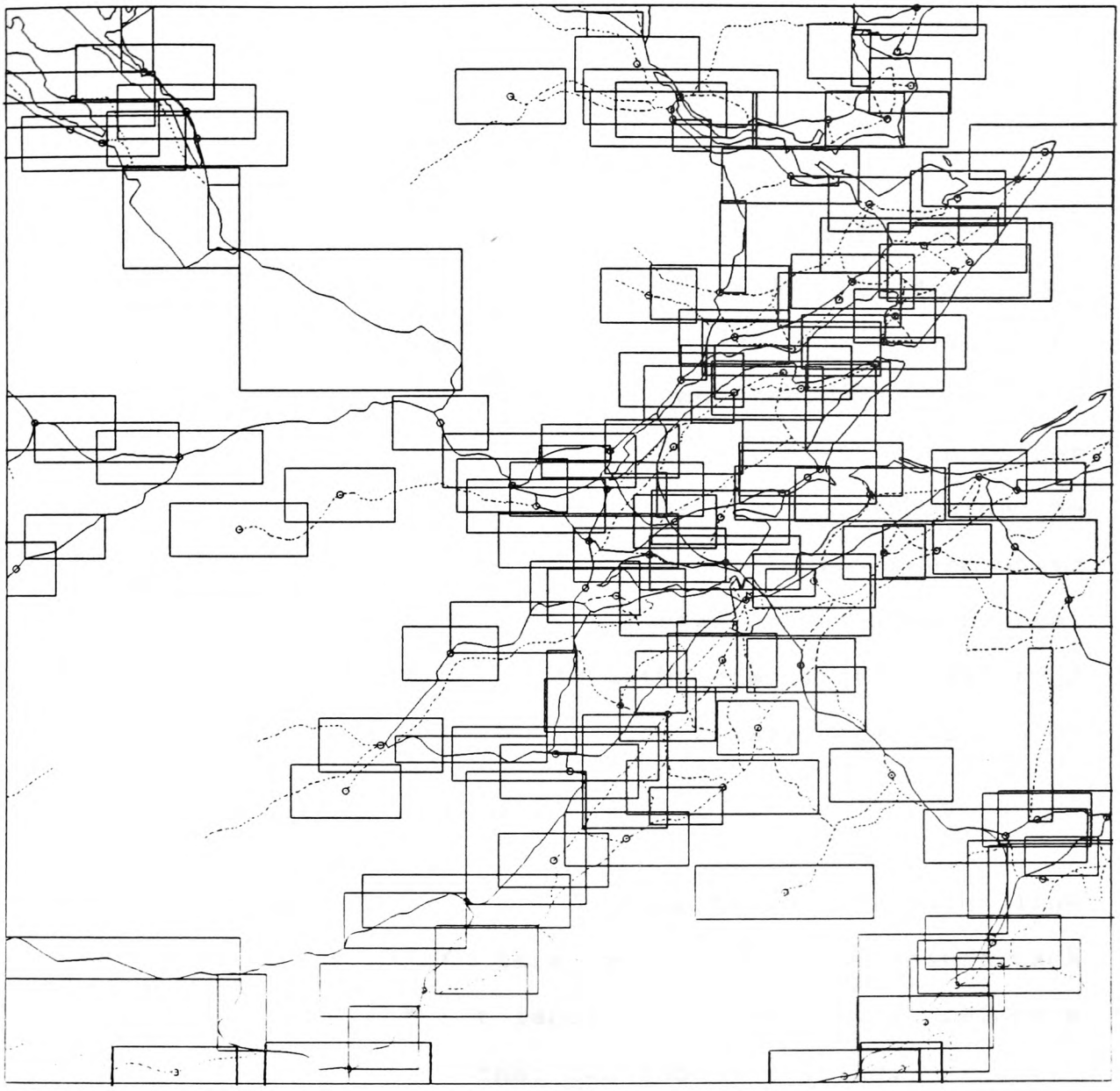


Fig 6.10 Plot of rectangles which bound region of containment for each label on grid square 208. This region is of a relatively low label density and yet overlaps between bounding rectangles suggest that the placement of a label at the bottom of the map could affect a placement at the top of the map. This is however very unlikely to occur in practice.

label now forms a horizontal window in which lines or corners of the smaller label can be tested for clipping and containment (Fig 6.11).

For a simple test of overlap, based upon the Cohen-Sutherland (Harrington, 1987) clipping algorithm, the space around the window is divided into 8 regions by the extension of window edges and each region is assigned a unique binary number code (Fig 6.12). Next, each corner of the second label is tested to see if it lies inside the window region, "0000", or the line between consecutive corners lie within opposite horizontal or vertical central regions of space. This can be determined if the result of a logical OR between the two region binary codes is a 3 or a 12 (Fig 6.13).

If this simple test for overlap fails then each line segment of the second label must be tested against each line segment of the first label (clipping window) before a conclusion can be drawn that two labels overlap.

6.6.4 LABEL:LABEL SEPARATION DISTANCE - RULE [6.13]

Although the label block heights and widths have been introduced to avoid labels overlapping or nearly touching each other, an additional rule can be used to encourage labels to separate further. This is achieved by specifying

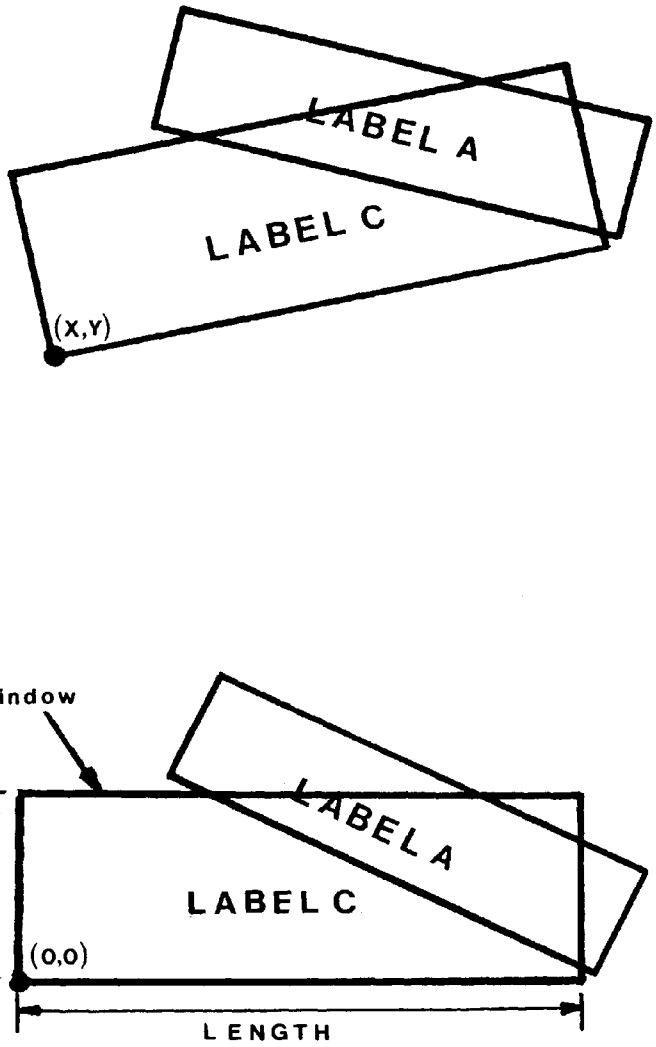


Fig 6.11 Rotation and translation of labels so that the larger label becomes the clipping window.

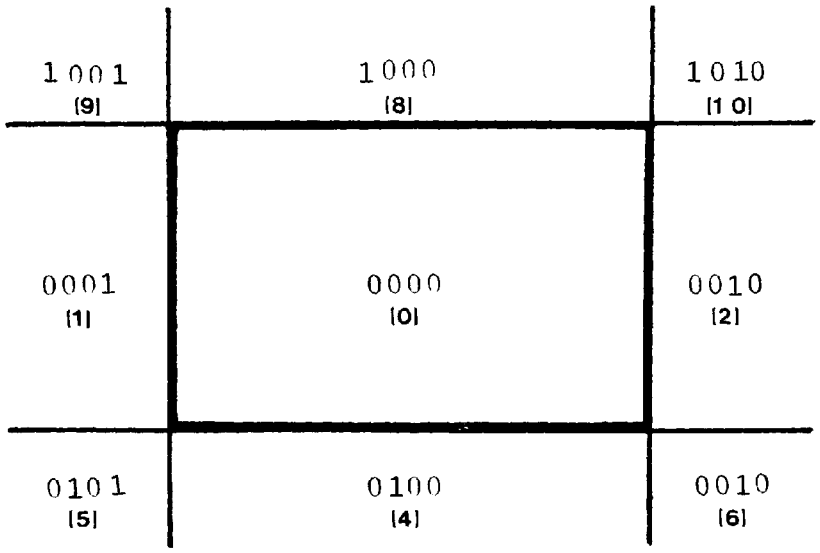


Fig 6.12 The reference system for the Cohen-Sutherland clipping algorithm.

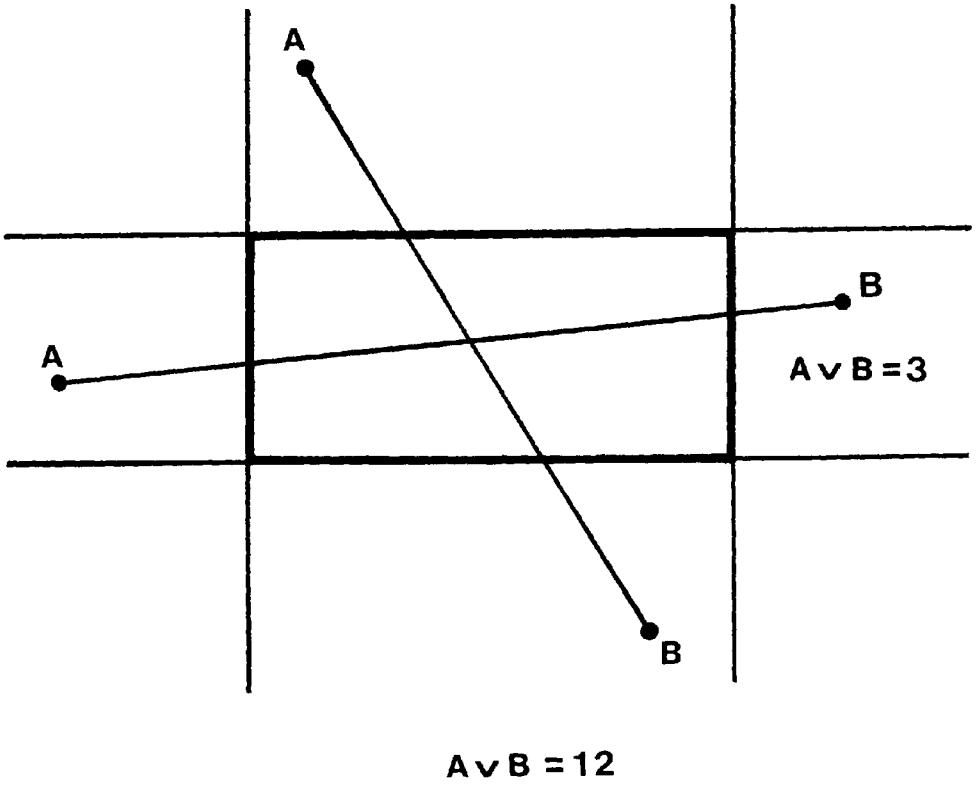


Fig 6.13 Detection of horizontal or vertical clipping.

a separation distance for each label which effectively adds a buffer zone around the label vertical and horizontal edges (Fig 6.14). The buffer zone can either be specified in terms of character block widths in which case they will vary with the feature code of the label, or alternatively, in terms of a fixed amount in metres on the ground.

Most labels are placed at least a buffer distance apart, but in the case of seriously conflicting labels, this specification can be relaxed in order to find a compromise solution. This is achieved by reducing the buffer separation if difficulties are experienced in placing a label.

Since line labels are usually at an angle to horizontal settlement labels, they can be placed almost touching other labels without actually appearing to be associated. Hence the label separation distance does not apply to line labels.

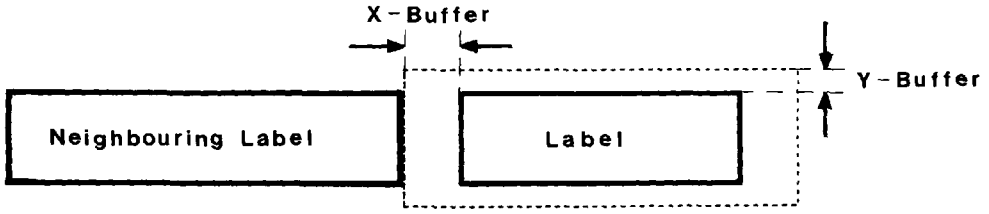


Fig 6.14 Enlargement of a label.

6.7 LABEL CONFLICT RESOLUTION

6.7.1 INTRODUCTION

The LABPOS strategy for label conflict resolution on the Route Planner map attempts primarily to place labels so as to avoid overlapping other labels. It also attempts to satisfy two secondary constraints, namely that label ambiguity is avoided and also that labels should preferably lie in free space or, failing that, over regions of low feature density as defined by the dense space threshold (Section 6.5.3).

The algorithm for solving the name placement problem in LABPOS involves three stages. Firstly to detect and attempt to resolve label conflicts over several iterations. Secondly to tidy up labels which have been moved during the first stage, but for which more preferred placements have become available again. Finally to identify and flag labels which either still remain in overlap or are too close together (conflict).

In the first stage, labels are initially placed in their most preferred positions and then, during each subsequent iteration, labels are tested to see if they are in conflict with other labels. If so, then the current label is tried in all its permitted positions, whilst keeping its neighbours fixed, until either a non-conflict

position is found, or there are no more positions to test. When testing positions, the first one to be tried is the current position of the label. This is the most likely position to test because during the previous iteration other labels in conflict may have been forced to move away thus freeing the current position. If this position fails then all the positions are tested in order of preference so as to favour more open positions.

When a label is found to be in conflict at every position, an overlap weight is determined for each position. This is normally the number of labels in conflict. However, when a label is "experiencing placement difficulty", as indicated by the number of placement attempts (Rule [6.14]), the weight of a position is re-defined as the sum of the number of attempts at placement of all the overlapping labels. If every position for the label is found to be in conflict with other labels, then the position at which the minimum overlap weight with respect to other labels occurred is selected. This position might not be ideal, but over several iterations the effect is to move labels outwards and away from regions of conflict. Also, as the iterations progress, the definition of label conflict applied to labels which are in persistent conflict is gradually relaxed to aid placement. This involves linearly decreasing the separation or buffer distance applied around the label, as described in section 6.6.4 (Rule

[6.13]), each time the label is found to be in conflict at all positions. In such situations the buffer distance continues to decrease until, after encountering N repeated placement attempts, it reaches zero and the label is deemed to have failed to be placed and is fixed in its current position for the remainder of the first stage of the conflict resolution algorithm (Rule [6.15]).

Once the iterative attempts at label conflict resolution have been completed, most labels will have been satisfactorily placed. However, there may be a few labels where the more preferred positions which were ruled out earlier could have become free again. Therefore all the labels are tested to see if any of them can be moved back into more preferred positions. When testing these positions, labels are enlarged to their original buffer distance so as to ensure that labels are placed a reasonable distance apart. If the position being tested is less preferred than the original label position, then the label remains in its original position.

6.7.2 LABEL WEIGHTING - RULE [6.14]

If a label is placed many times during the iterative name placement process, then this indicates that it is experiencing some difficulty. If a label overlaps other labels at all of its available positions then, after a

user specified number of such occurrences, it is weighted so as to avoid overlapping other labels which are experiencing similar difficulties in favour of those which have been moved least and which have a greater freedom of movement.

6.7.3 NUMBER OF CONFLICT RESOLUTIONS BEFORE FAILING - RULE [6.15]

If a label regularly fails to be placed in a non-conflict position, despite the application of weighting, then it is reasonable to assume that the label will only have its conflict resolved by fixing it at its most preferred position with respect to other labels and forcing its neighbours to try new positions more rigorously. The user can specify how many attempts at placement are permitted before LABPOS decides to fix the position of this label. The label then remains fixed throughout the remainder of the first stage of the conflict resolution algorithm.

6.7.4.1 HIGH LEVEL PSEUDO CODE FOR CONFLICT RESOLUTION

```
place all labels in their most preferred positions

dowhile  conflicts to be resolved
   detect and resolve label conflicts
endwhile

tidy-up label placement

flag any un-resolved label overlaps or potential ambiguity
```

6.7.4.2 LOW LEVEL PSEUDO CODE FOR CONFLICT RESOLUTION

```
(* detect and resolve label conflicts *)
initialise each label to its most preferred position (* current *)

initialise conflicts_present to true

dowhile conflicts_present
   conflicts_present:= false
   for all labels
      if label has NOT been fixed in position (* Rule [6.15] *) then
         compute label dimensions & buffer size
         (* buffer is reduced according to No. of attempts
          to resolve conflicts for label *)
         initialise min_no_of_overlaps to 9999
         initialise no_of_overlaps to 9999

         dowhile no_of_overlaps > 0 and other positions are available
            initialise no_of_overlaps to 0
            (* if not original position *)
            if first position/time around this loop then
              use label's current position
            else
              compute new label position
              (* Working in order of preferred positions *)
            endif
            if label centre lies inside 100x100km window then
               for all labels in potential overlap with current label
                  test for overlap
                  if overlap then
                     if No of placement attempts of current
                       label suggest placement difficulties then
                        (* Rule [6.14] *)
                        no_of_overlaps := no_of_overlaps +
                                    No of placement attempts
                                    of other label
                     else
                        increment no_of_overlaps by one
                     endif
                  endif
               endfor (* all labels in potential overlap *)
               if no_of_overlaps < min_no_of_overlaps then
                 min_no_of_overlaps := no_of_overlaps
                 min_no_of_overlaps_pos := label position
               endif
            endif (* label centre *)
            if first time round this loop then
               select most preferred position
            else
               select next most preferred position
            endif
         endwhile
         place label at position min_no_of_overlaps_pos
         if label regularly fails to have its conflicts resolved
              (* Placement attempts for label have failed a
              user specified number of times [6.15] *) then
            fix label in current position
         endif
         if min_no_of_overlaps > 0 then
            conflicts_present := true
         endif
      endif
   next label
endwhile
```

```
(* tidy up labels placement *)

for all labels
   compute label dimensions + buffer
   initialise position to most preferred position
   initialise overlap to true
   dowhile position preference > original position preference
      AND overlap
      overlap := false
      dowhile potentially overlapping labels to investigate
        AND not(overlap)
        test for overlap
      endwhile
      if overlap then
        increment position
      endif
   endwhile
   if not(overlap) AND not(position = original position) then
     update label record with new position
   endif
endfor


(* flag any un-resolved label overlaps or potential ambiguity *)

for all labels
   compute label dimensions + buffer
   initialise overlap to false
   dowhile potentially overlapping labels to investigate
        AND not(overlap)
      test for overlap
   enddo
   if overlap then
      update label record with overlap flag
   endif
endfor
```

6.8 OUTPUT OF LABEL POSITIONS

The label records are output, using the same record structure as "LABEL1", to a file prefixed by the abbreviation "LAB" and suffixed by the three digit 100x100km sq. library number. For purposes of simple graphical output the file is sequential. It contains all the information required for necessary interaction with the Ordnance Survey 1:625000 database, and also contains a listing of all possible positions and potential overlaps for each label. These may be of assistance in the optimization of some of the user defined rules and specifications which will be discussed in the next section.

To output the label information on a map, it is first necessary to draw the background map. After plotting this, each label is stepped through in the "LABenn" file and the offsets and orientations of each letter making up the name are calculated. The letter sizes and placings are computed with the aid of the "FEAT_RP" data file described in section 6.2 and indexed by the feature code field in the "LABenn" data. Occasionally, the standard character sizes assigned to labels on a feature code basis can be overidden by an enlargement factor for a particular name in the Ordnance Survey database. If enlargements have been made to some of the names, then these can be identified from the output label data by comparing label block

heights, as defined in both the "LABenn" and the "FEAT_RP" files, and corrections applied by scaling the default label block and letter dimensions to those of the enlarged label.

The letter characteristics can be adjusted by calling the appropriate routines in graphics packages such as GINO-F or GKS which allow both letter size and orientation to be adjusted prior to plotting. However, in the examples produced for this chapter, the specified letter lower case height ("FEAT_RP" file) and descender height have been ignored in preference to the default graphics values. If necessary, modifications can be made to the type script or colour at the user's discretion.

For practical use by the Ordnance Survey it will be necessary to interact with the Route Planner map database in order to edit labels which have not been placed successfully. This can be achieved using the feature serial number and name identification number fields in "LABenn" which relate directly to the same fields in the Route Planner database. If the name has been split and modified by LABPOS, indicated by the SPLIT filed, then the label string will consist of an even number of letters and may be padded out with spaces.

The last field in the "LABEnn" file gives the user an indication of which labels maybe in conflict. This is in the context of labels which either overlap each other or are very closely placed and should be drawn to the attention of the user. It was not the intention of the research to develop such an interactive system, but the information has been made available. In fact the Ordnance Survey already has a graphical interactive system for manually placing names (Hadley, 1986), which it might be possible to adapt for this purpose.

6.9 LABPOS RESULTS

6.9.1 INTRODUCTION

This section of the chapter demonstrates how user controllable name placement rules and specifications affect both the optimization of the system and the map design. Optimization is concerned with changing rules and specifications so that the program works both quickly and minimises label conflict but not at the expense of ignoring too much underlying detail. The map design defines how the names are presented on the map, for instance their size, preferred placement positions and how to split multi-worded labels. Optimization of the system and map design are interelated. For instance a map design involving big labels increases the risk of label conflicts and increases the program run time. To achieve the best results with LABPOS, map design and system optimization should be achieved through compromise.

If LABPOS is to be used on a variety of different grid squares from the Route Planner map then ideally the specifications and rules should be optimized to cope with each particular locality. In practice if more than one grid square is to be labelled this will take too long and so this chapter describes how to optimize the user definable rules and specifications for grid squares in general and their effects on map design.

The effects will be demonstrated using three grid squares which are typical of the range of densities encountered on the map. However because of the large quantity of detail present, ellipses will be used to highlight examples of where conflicts have arisen or a change in label arrangement has taken place. When interpreting the results, the reader should judge the quality of the placement according to the number of unresolved label conflicts, the presence of ambiguously placed labels and placement with respect to underlying detail. Comparisons can made with the original Route Planner maps (Fig 2.18, 2.19 and 2.20) however these original maps contain a slightly higher cartographic feature and label density than in the Ordnance Survey Route Planner map database. The three figure numbers used to refer to the maps in this chapter correspond to database library grid squares numbers (Fig 2.17).

6.9.2 NAME PLACEMENT OPTIMIZATION

6.9.2.1 INTRODUCTION

To speed up the name placement process the chances of resolving label conflicts must be improved and the number of name placement iterations kept as small as is practicable. Because there are so many rules/specifications that can be adjusted, it is best to

optimize these one at a time, starting with those considered to have the most influence and also those whose effects are easily hidden by other rules. However before this can be done, some reasonable starting values for the user definable rules and specifications must be provided. These are listed in Tables 6.4 and 6.5. The preferred ordering of point labels is based upon the findings of Section 2.3 (Appendix 1, Table 1)and the ordering of line label positions is designed to favour the centre of lines. The buffer or separation distance between labels has been set to zero so that when a label conflict is detected this corresponds to a label overlap (See section 6.6.1 for definitions).

The first user controllable rule (specification) that will be investigated is the raster size which can have a significant effect on the total number of label positions available, the number of conflicts that result and hence the run time of LABPOS. This will then be followed by the number of placement attempts allowed before weighting or fixing the label, which also affects the run time. Once this has been established, the dense space threshold will be investigated which has a major effect on the number of label positions available. Finally, but with a lesser effect, the label splitting threshold, the point label radius of proximity and the label separation or buffer distance will be investigated.

Table 6.4 Original settings for the "RULE" file.

Raster Size	512	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Dense space threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	3
1st preferred line position	8	Attempts before fixing label	6

Table 6.5 Original settings for non-zero records in the "FEAT_RP" file.

Feat. Code	Token Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n	1600	3	0	0	0	0	0	0
1	mway_service_ltd_e	1600	3	0	0	0	0	0	0
2	mway_service_ltd_s	1600	3	0	0	0	0	0	0
3	mway_service_ltd_w	1600	3	0	0	0	0	0	0
20	motorway	600	3	810	625	540	202	1313	900
21	motorway_junction	1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs	1600	3	0	0	0	0	0	0
23	mway_service_area	1600	3	0	0	0	0	0	0
30	primary_route_s_c	450	1	0	0	0	0	0	0
31	primary_route_d_c	550	2	810	625	540	202	1250	800
32	prim_route_narrow	400	1	810	625	540	202	1250	800
35	prim_route_srv_area	1600	1	0	0	0	0	0	0
40	main_road_s_c	300	1	810	625	540	202	1250	800
41	main_road_d_c	550	2	810	625	540	202	1250	800
42	main_road_narrow	300	1	810	625	540	202	1250	800
49	b_road	125	0	600	438	400	150	625	438
50	other_road	125	0	0	0	0	0	0	0
61	city	5000	4	1075	750	720	269	1375	1040
62	large_town	3000	4	1075	688	720	269	1312	938
63	village_on_prim_rt	1500	4	850	600	500	188	1125	625
64	town_on_prim_route	1500	4	1075	688	720	269	1312	888
65	small_town_village	1500	4	850	600	500	188	1125	625
75	airport	1450	1	900	700	500	188	1100	900

Because many of the rules are interdependant, optimization should be an iterative process and adjustment repeated until no significant improvement is noticed. In practice this could take several days and so for the purposes of this demonstration, only one optimization attempt will be made.

6.9.2.2 RASTER SIZE

The alteration of raster size has a significant effect on LABPOS in that the larger the size of pixel, the less pixels there are to count and the faster the system should work (Rule [6.3]). The graph shown in Fig 6.15 of Central Processing Unit (CPU) time on a VAX 11/785 mini-computer confirms this but the effect appears to be less prevalent for grid squares of lower label density. Unfortunately CPU time is not a totally reliable measure of the amount of processing that a job is undergoing on the computer due to page faults caused by other users on the system. This explains much of the noise present in the graphs, the data for which was generated at different times of the day.

A reliable illustration of the effect of varying raster size is given in Fig 6.16 which shows how the total number of positions available for all the labels gently decreases until the raster size falls below 150 x 150 at

which point the rate of fall increases noticeably. Similarly in Fig 6.17 the number of unresolved label conflicts increase rapidly for a raster size below 150 x 150. It is interesting to note that several bumps are visible in Fig 6.16 which are present for all three library square curves. This is a consequence of an increase in raster resolution where a feature changes from being N pixels wide to N+1 pixels and so increases the number of pixels that are counted at different label positions when examining underlying features.

A raster size of 350 x 350, with a corresponding pixel size of 0.46mm or approximately a third of the smallest letter height, has been selected as a reasonable compromise between CPU time, the number of positions available and resulting conflicts. As a comparison to illustrate the effect of different raster sizes, Fig 6.18 has been produced using a raster size of just 100 x 100 or a pixel size of 1.6mm and contains approximately two thirds as many unresolved label conflicts as Fig 6.19 which was produced using the selected raster size. Although Fig 6.19 contains a few new label conflicts, particularly in the "KINGSTON UPON THAMES" area, its placements are generally of a higher quality than in Fig 6.18.

Table 6.6 Statistics for Raster Optimization.

Raster Size	Library Grid Square 201				Library Grid Square 301				Library Grid Square 501			
	Itera-tions	Posi-tions	Over-laps	CPU sec.	Itera-tions	Posi-tions	Over-laps	CPU sec.	Itera-tions	Posi-tions	Over-laps	CPU sec.
0	-	165	165	-	-	339	339	-	-	284	284	
25	7	658	43	184	14	947	102	454	7	713	105	327
50	7	1243	17	182	13	2134	78	607	11	1586	55	425
75	10	1158	20	219	10	2136	74	572	10	1702	54	411
100	9	1388	6	208	11	2709	58	534	11	2215	44	440
125	11	1558	8	229	13	3246	38	557	11	2517	30	431
150	14	1650	8	276	11	3500	27	615	11	2709	25	434
175	9	1705	8	226	10	3565	37	580	11	2738	24	432
200	7	1543	13	200	12	3226	36	540	11	2494	33	423
250	11	1638	8	255	12	3359	37	594	11	2636	28	446
300	11	1718	11	255	13	3567	28	584	11	2750	20	438
350	11	1646	11	255	13	3477	29	613	11	2667	27	452
400	11	1696	6	251	13	3657	35	630	11	2792	19	454
450	7	1760	8	202	13	3794	27	631	11	2894	10	463
500	11	1718	8	260	13	3664	32	648	11	2815	16	461
550	11	1748	9	257	11	3740	29	597	11	2827	16	472
600	11	1703	6	262	11	3689	29	612	11	2826	16	485
650	9	1734	4	247	13	3757	26	647	11	2893	12	494
700	11	1751	10	281	13	3762	28	679	11	2903	14	505

Fig 6.15 Graph of CPU time versus raster size.

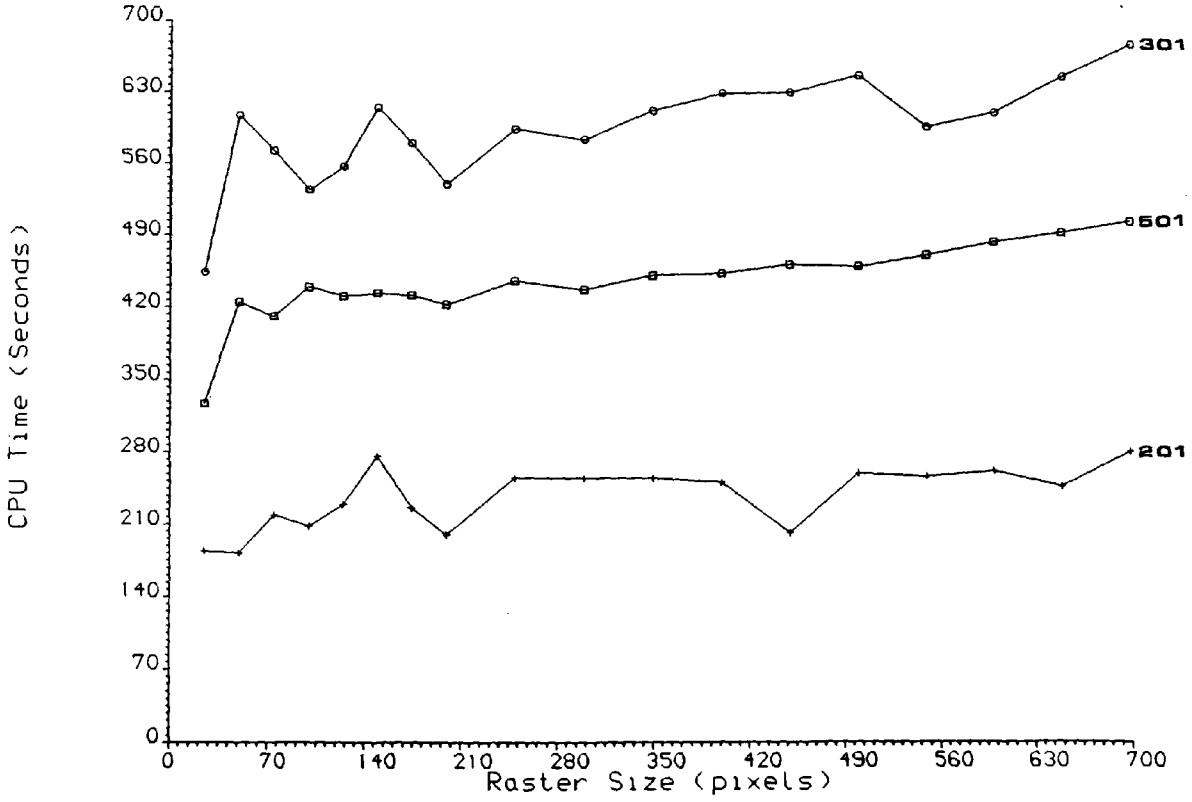


Fig 6.16 Graph of total number of positions available to labels versus raster size.

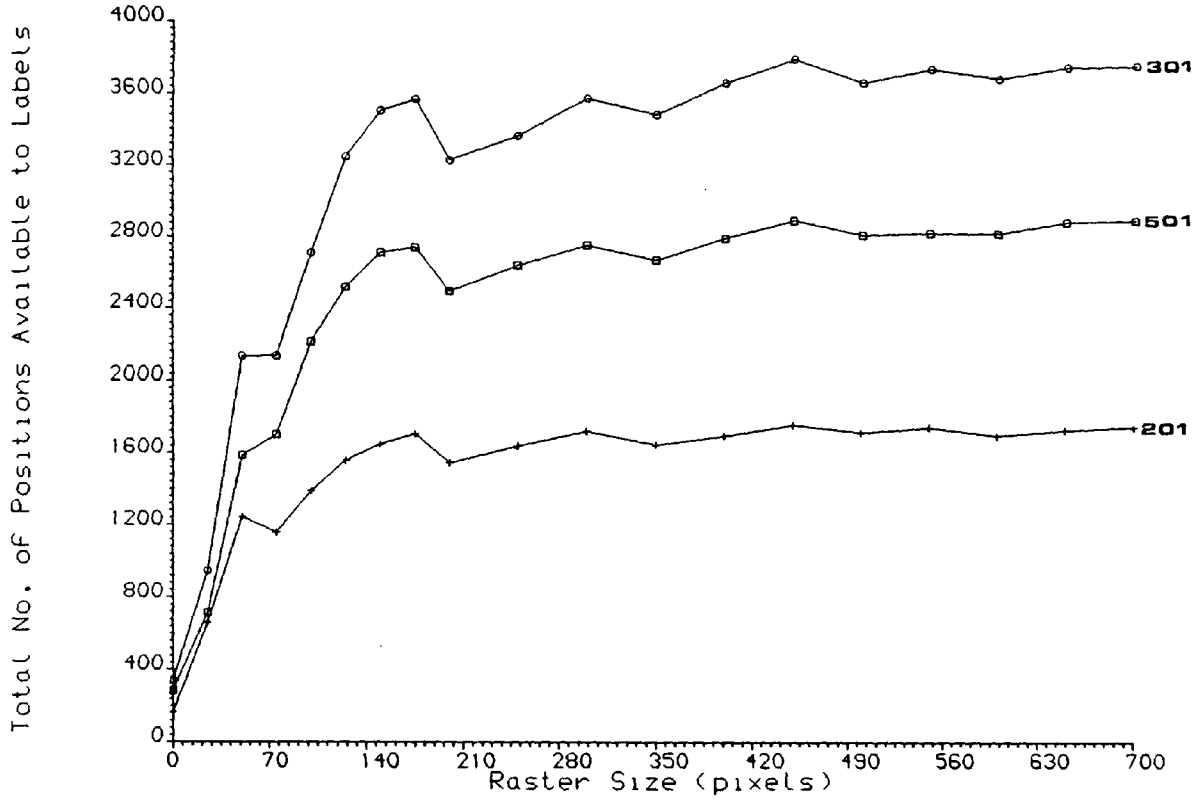
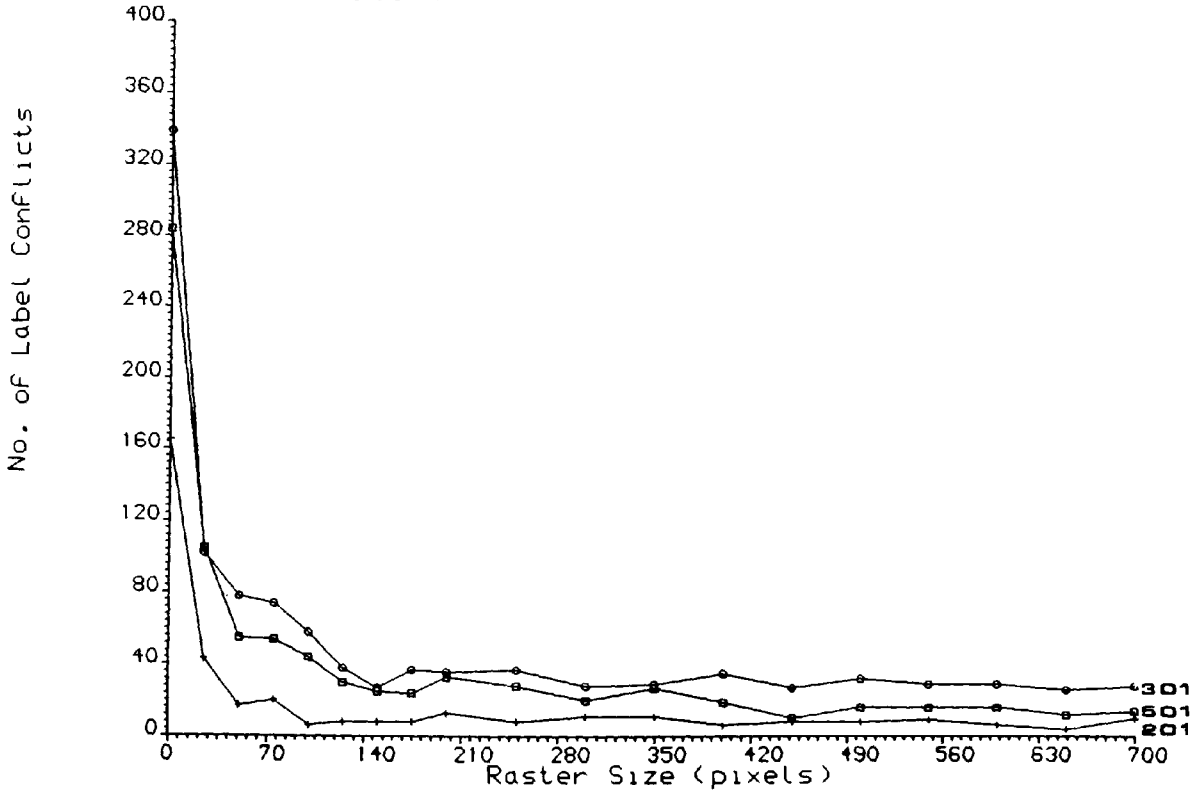
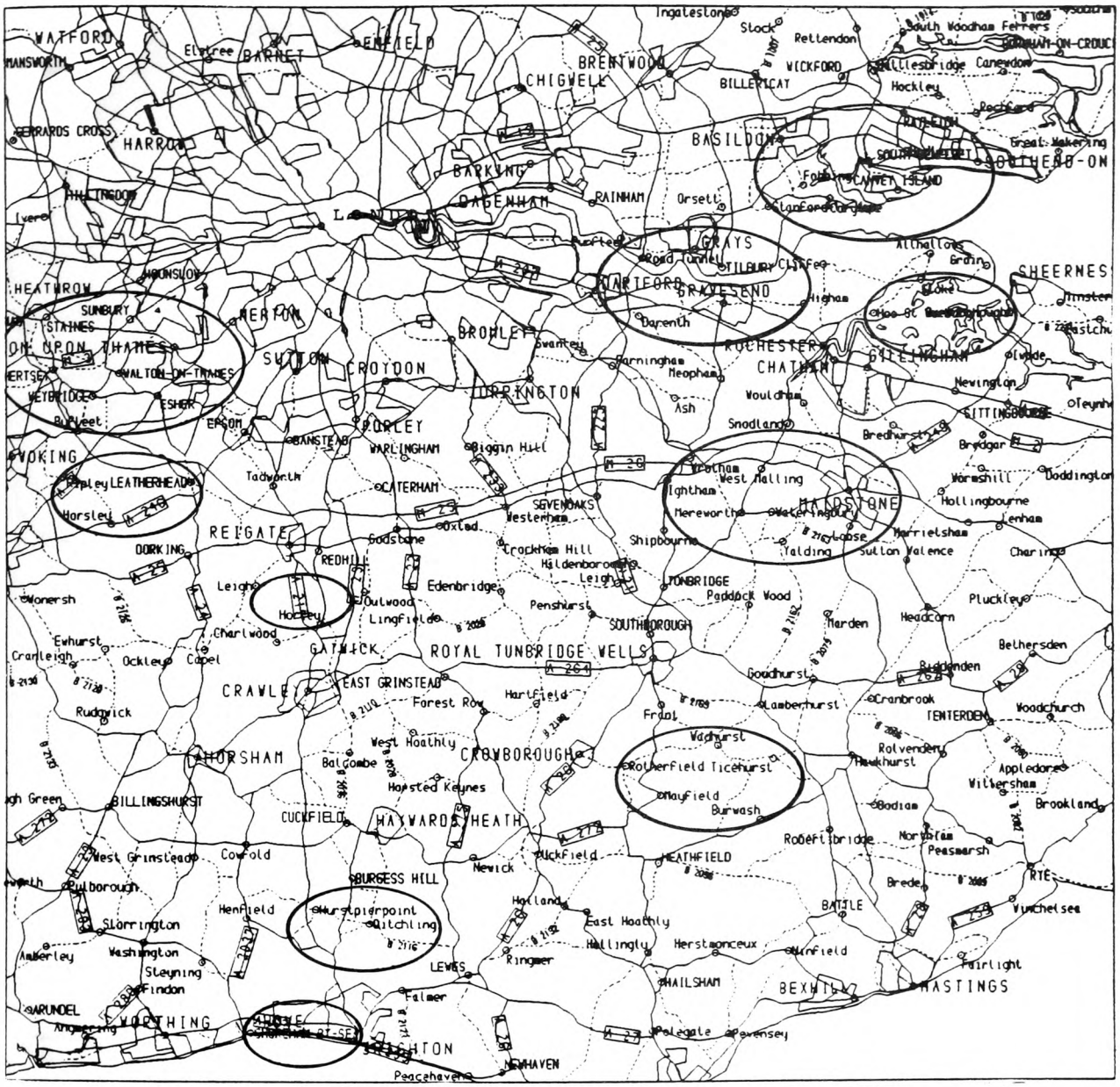


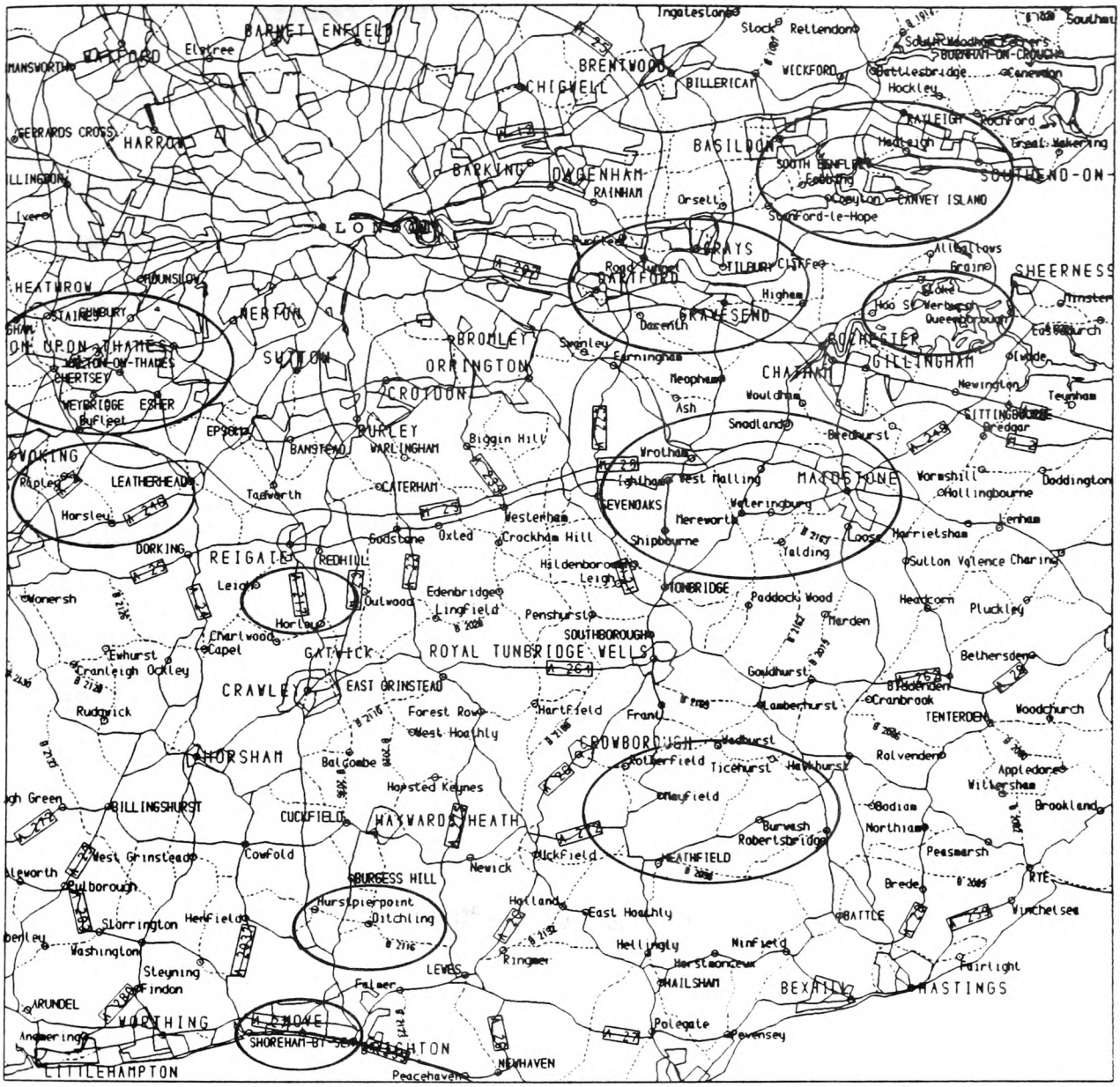
Fig 6.17 Graph of total number of label conflicts versus raster size.





Raster Size	100	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	3
1st preferred line position	8	Attempts before fixing label	6

Fig 6.18 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Raster size 100 x 100.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	3
1st preferred line position	8	Attempts before fixing label	6

Fig 6.19 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Raster size 350 x 350.

6.9.2.3 PLACEMENT ATTEMPTS BEFORE WEIGHTING OR FIXING A LABEL

Using the above raster size, the use of different numbers of placement attempts before weighting a label or fixing it in position were investigated (Table 6.7). It is apparent that the number of placement attempts before fixing a label makes little difference to the outcome except for the total number of iterations required by the program and hence CPU time. The weighting also appears to have relatively little effect except that the number of iterations required is generally less if weighting is applied as soon as possible. If weighting is not applied until the label is fixed then the number of conflicts that result is usually greater.

The application of rules [6.14] and [6.15], to labels which are experiencing difficulty in placement over several iterations, is heuristically sound. Unfortunately the proportion of labels experiencing difficulty in placement is low and so the results of Table 6.7 are swamped by other factors. Also rule [6.14] takes several iterations to come into effect and so will only be of use when LABPOS has undergone a large number of iterations.

In view of the CPU time factor and the need to start weighting as soon as possible, it was decided to allow for two placement attempts before weighting and five before

Table 6.7 Statistics for finding optimum values for the number of placement attempts before weighting or fixing a label.

Weight No.	Fix No.	Library Grid Square 201		Library Grid Square 301		Library Grid Square 501	
		Conflicts	Iterations	Conflicts	Iterations	Conflicts	Iterations
2	2	11	3	36	3	28	3
2	3	11	7	29	7	27	7
3	3	11	4	36	4	28	4
2	4	11	8	29	10	27	8
3	4	11	9	29	9	27	9
4	4	11	5	36	5	28	5
2	5	8	9	29	11	27	9
3	5	10	11	29	12	27	10
4	5	11	11	29	11	27	11
5	5	11	6	36	6	28	6
2	6	9	8	29	13	27	11
3	6	11	11	29	13	27	11
4	6	11	12	29	14	27	12
5	6	11	13	29	13	27	13
6	6	11	7	36	7	28	7
2	7	9	8	29	14	27	12
3	7	9	10	29	14	27	12
4	7	11	13	29	15	27	13
5	7	11	14	29	16	27	14
6	7	11	15	29	15	27	15
7	7	11	8	36	8	28	8
2	8	9	9	29	20	27	14
3	8	9	10	29	15	27	14
4	8	11	14	29	16	27	14
5	8	11	15	29	17	27	15
6	8	11	16	29	18	27	16
7	8	11	17	29	17	27	17
8	8	11	9	36	9	28	9

fixing the label in position. Fig 6.20 and 6.21 illustrate the effects of not applying weighting and weighting early. It is apparent that there is a slight improvement when weighting is applied early.

6.9.2.4 DENSE SPACE THRESHOLD

Although the dense space threshold (Section 6.5.3) is intended primarily for map design purposes, it also affects the total number of label positions and consequently conflicts (Rule [6.12]). Fig 6.22 illustrates that the number of available positions increases as the dense space threshold is increased. However when the threshold exceeds 70%, the improvements in label conflicts becomes small (Fig 6.23).

Figs 6.24, 6.25 and 6.26 clearly reveal that the number of label conflicts falls with an increase in dense space threshold but at the expense of obscuring underlying detail. Although a dense space threshold of 70% would seem reasonable to adopt, a value of 50% has been selected so as to ensure a plentiful number of available positions but not at the expense of obscuring an unacceptable amount of underlying detail.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	5
1st preferred line position	8	Attempts before fixing label	5

Fig 6.20 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - No label weighting.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.21 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Label weighting starts on 2nd label placement attempt.

Table 6.8 Statistics for dense space threshold Optimization.

Dense Space Threshold	Library Grid Square 201			Library Grid Sqaure 301			Library Grid Square 501		
	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions
0%	705	47	10	1543	118	9	1146	97	9
10%	1153	25	6	2318	73	12	1724	63	6
20%	1390	15	8	2808	48	9	2091	47	9
30%	1507	13	8	3108	37	9	2372	36	11
40%	1570	11	8	3313	32	12	2542	26	9
50%	1646	9	8	3477	29	11	2667	27	9
60%	1676	9	8	3609	25	11	2776	21	9
70%	1690	6	8	3678	19	11	2850	19	9
80%	1696	6	8	3721	17	11	2906	19	9
90%	1698	6	8	3742	17	11	2946	19	9
100%	1779	6	8	3886	17	11	3088	17	9

Fig 6.22 Graph of total number of positions available to labels versus dense space threshold.

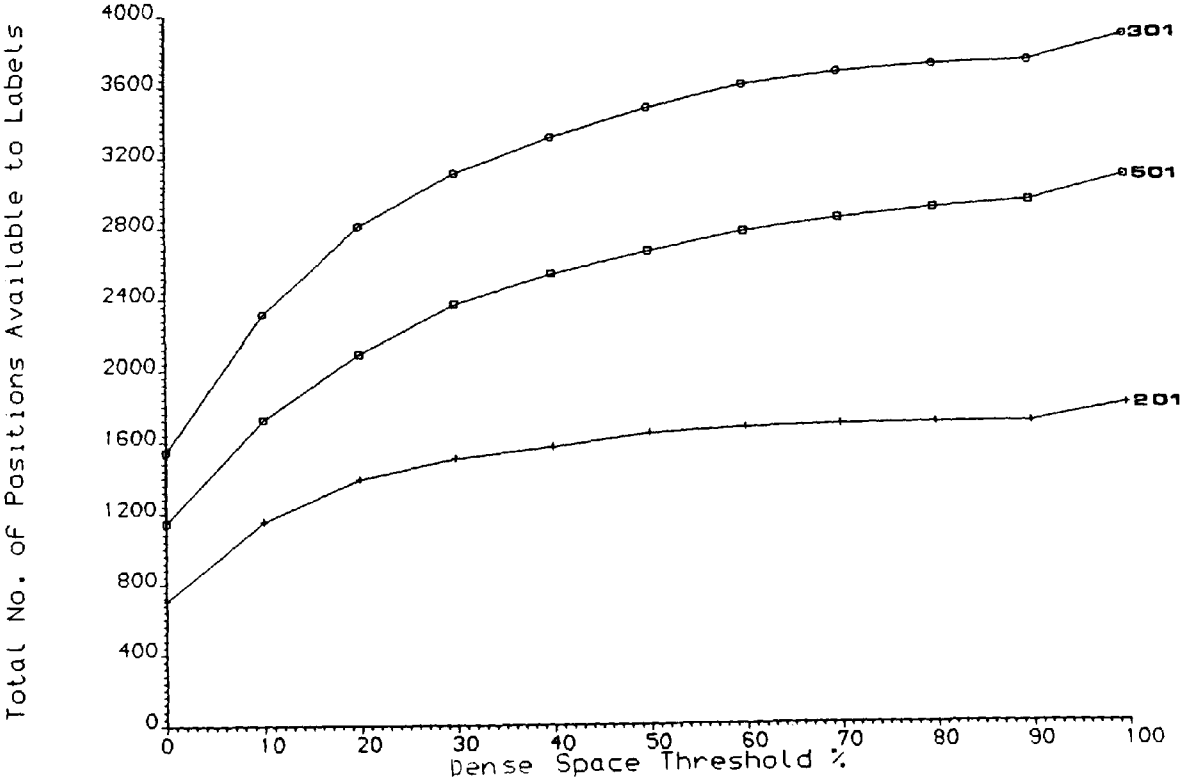
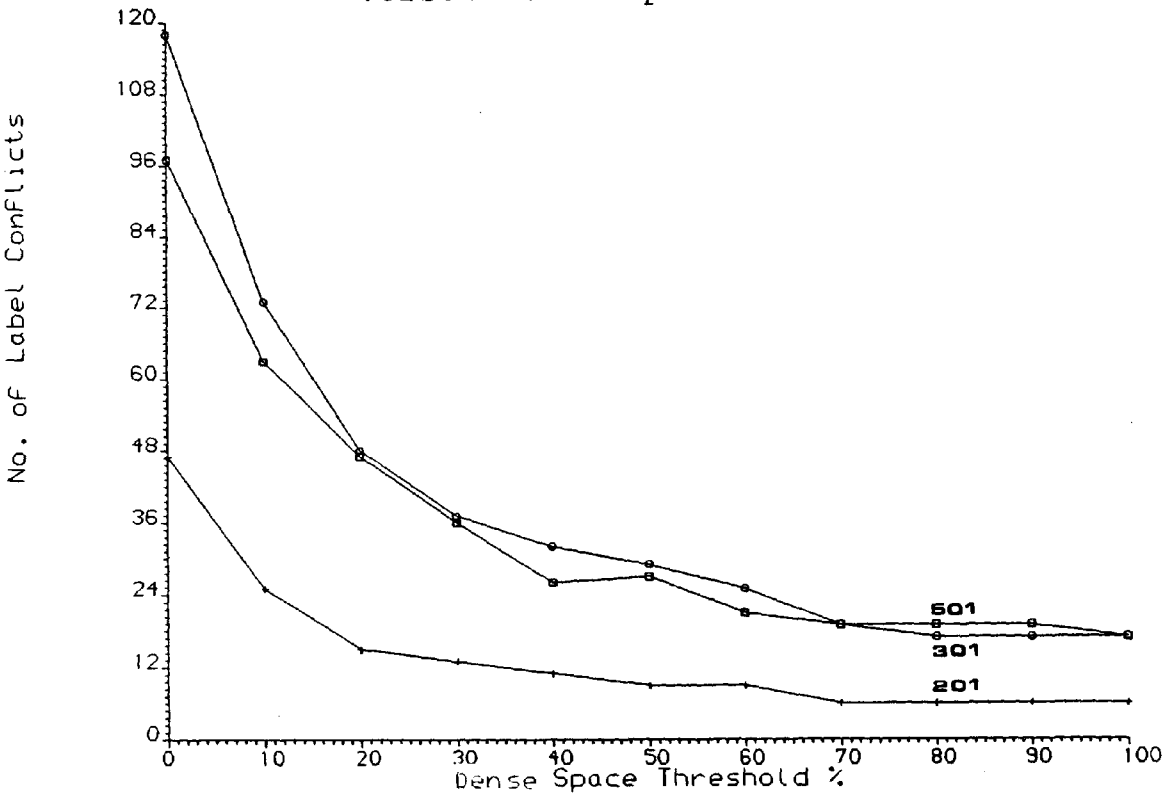
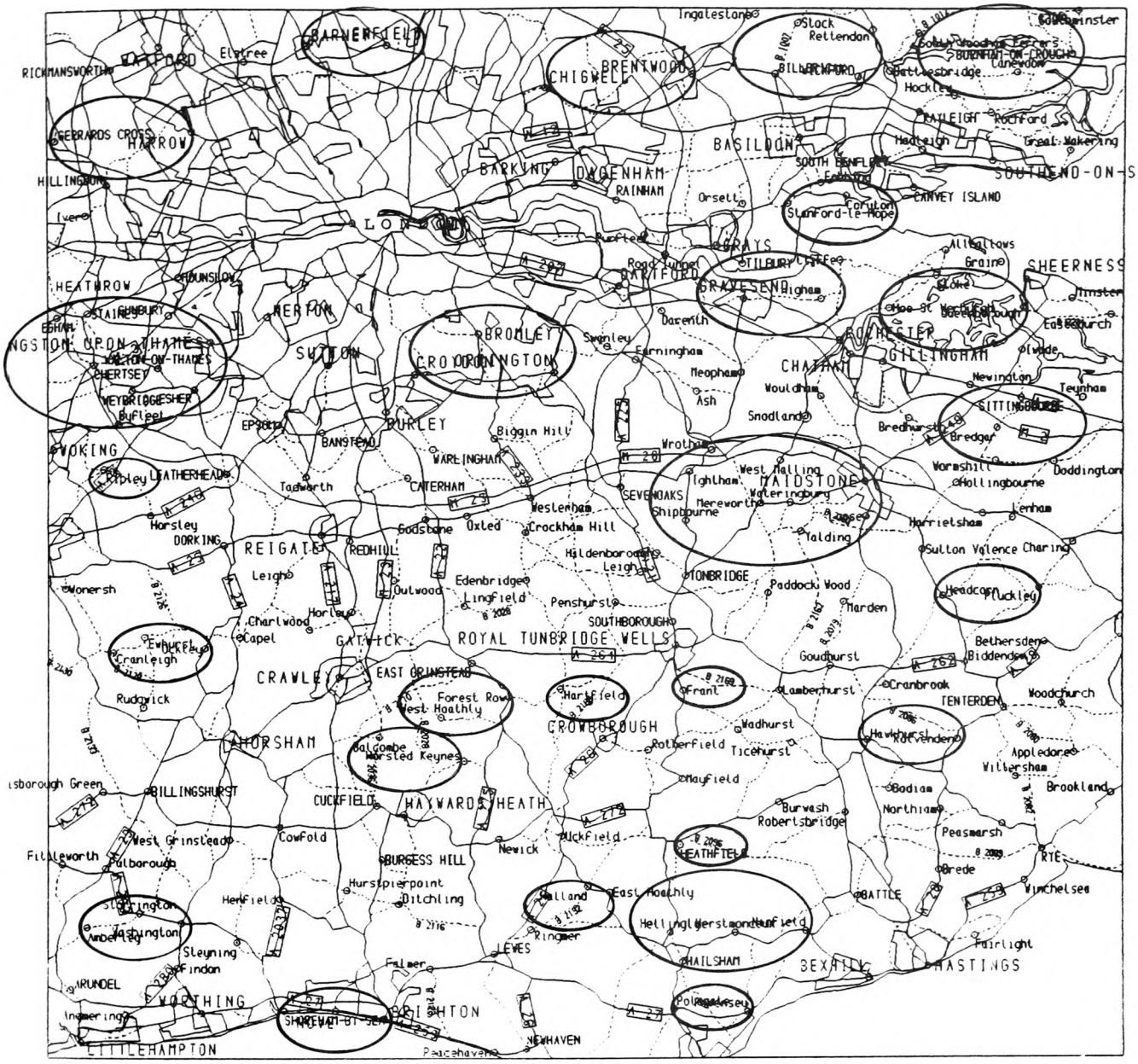


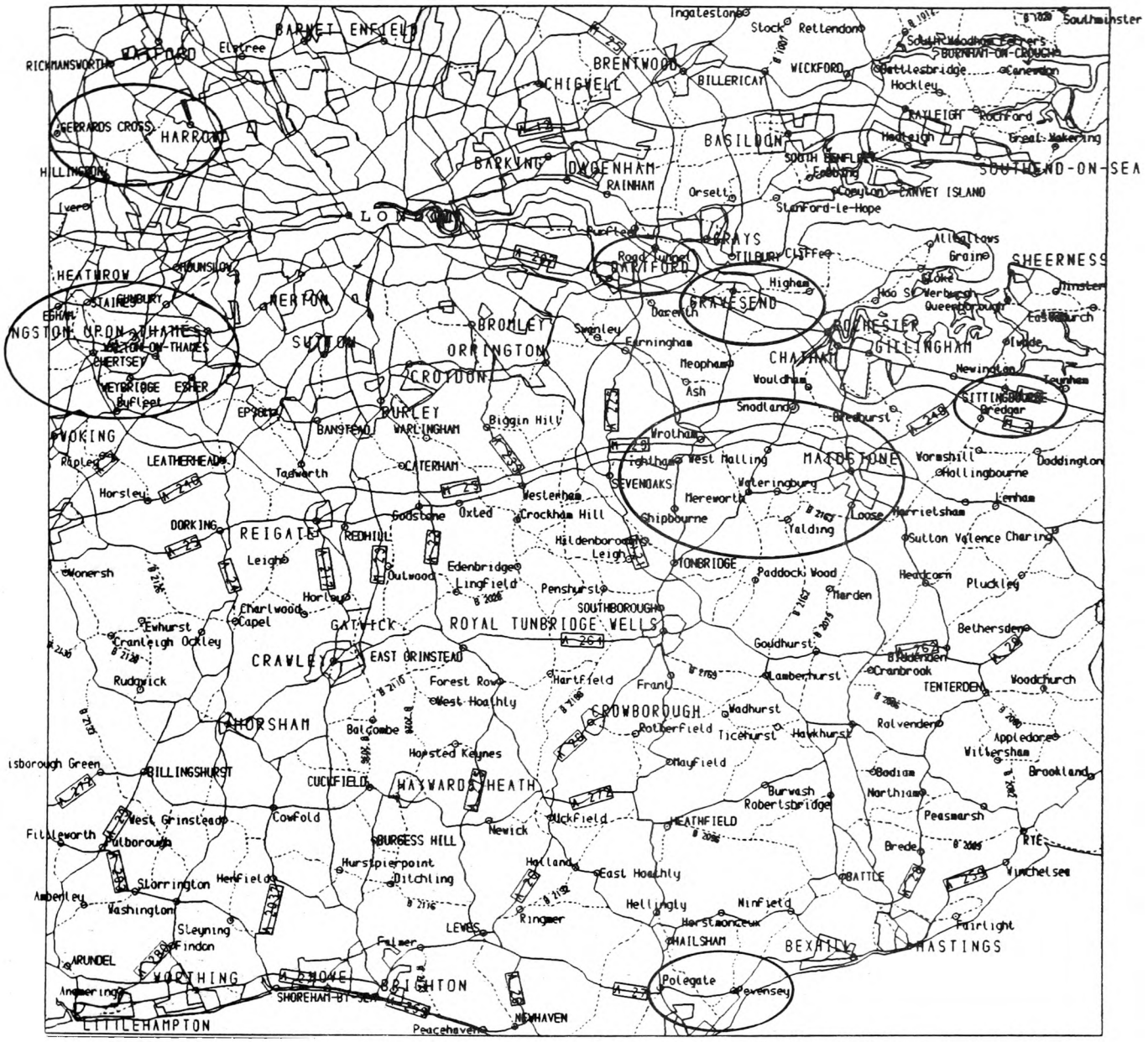
Fig 6.23 Graph of total number of label conflicts versus dense space threshold.





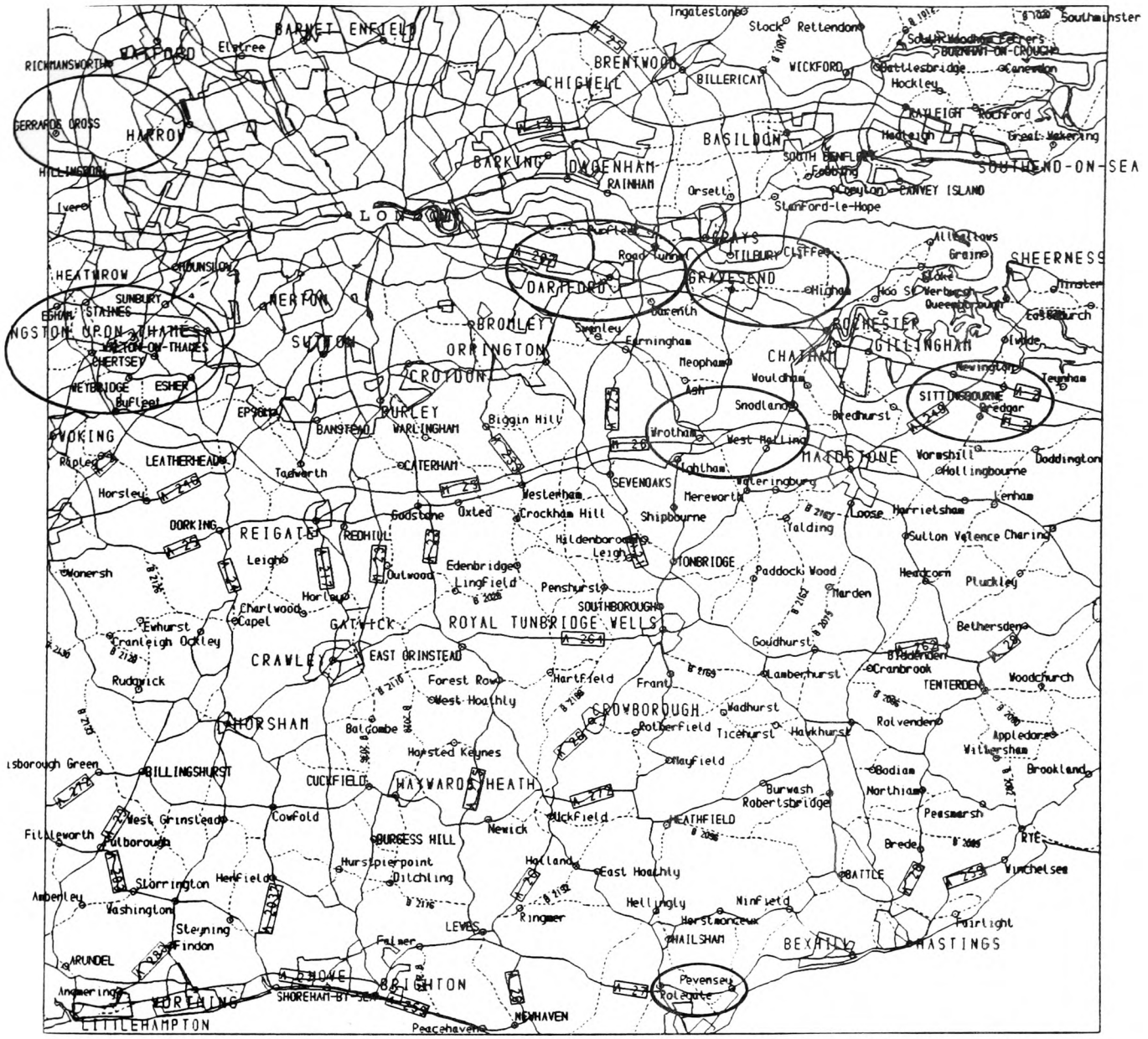
Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	0
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.24 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Dense space threshold 0%.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.25 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Dense space threshold 50%.


BARNET ENFIELD
WATFORD
RICKMANSWORTH
Elstree
BRENTWOOD
CHIGWELL
BILLERICAY
WICKFORD
Ingatestone
Stock
Rettendon
Battlesbridge
Hockley
RAYLEIGH
Rochford
Great Wakering
SOUTHEND-ON-SEA
GERRARDS CROSS
HARROW
HILLINGDON
BARKING
DAGENHAM
RAINHAM
BASILDON
Orsett
Stanford-le-Hope
Canvey Island
Iver
LONDON
Purfleet
Road Tunnel
GRAYS
TILBURY
CLIFFE
GRAVESEND
DARTFORD
Higham
Cobham
Meopham
Swanley
Farningham
Dunton Green
Sevenoaks
Otford
Shoreham
Eynsford
Crockham Hill
Westerham
Oxted
Godstone
Caterham
Tadworth
Biggin Hill
Warlingham
Banstead
Epsom
Ewell
Cheam
Sutton
Merton
Kingston upon Thames
Surbiton
Chertsey
Weybridge
Esher
Byfleet
Walton-on-Thames
Staines
Sunbury
Hounslow
Heathrow
Ripley
Horsley
Leatherhead
Woking
Dorking
Reigate
Redhill
Merstham
Coulsdon
Croydon
Purley
Bromley
Orpington
Swanley
Farningham
Meopham
Ash
Wouldham
Snodland
Aylesford
Maidstone
Chatham
Rochester
Gillingham
Strood
Higham
Cobham
Northfleet
Gravesend
Sittingbourne
Newington
Teynham
Faversham
Boughton
Ospringe
Eastchurch
Sheerness
Minster
Queenborough
Leysdown
Warden Point
Harrietsham
Lenham
Charing
Sutton Valence
Loose
Detling
Bearsted
Hollingbourne
Boughton Monchelsea
Staplehurst
Headcorn
Pluckley
Bethersden
Tenterden
Woodchurch
Appledore
Ashford
Wittersham
Brookland
Rye
Winchelsea
Hastings
Fairlight
Bexhill
Battle
Crowhurst
Sedlescombe
Robertsbridge
Burwash
Northiam
Peasmarsh
Rolvenden
Hawkhurst
Cranbrook
Goudhurst
Lamberhurst
Paddock Wood
Marden
Tonbridge
Hildenborough
Leigh
Penshurst
Edenbridge
Lingfield
Oxted
Godstone
Caterham
Tadworth
Warlingham
Biggin Hill

6.9.2.5 LABEL SPLITTING THRESHOLD

Because most labels are point labels and consequently horizontally aligned, short labels should stand a better chance of placement than long labels (Rule [6.8]). Table 6.9 and Fig 6.27 illustrate the number of available positions is slightly greater the more often labels are allowed to be split. Unfortunately the number of label conflicts does not appear to follow any particular trend. In view of this a label splitting threshold (Section 6.4.4.3) of 20% has been selected since the number of label conflicts is at a minimum in all three grid squares examined for this value. Fig 6.28 was produced with the recommended splitting threshold and appears to be a slight improvement over Fig 6.29 which was produced with a splitting threshold of 60%.

6.9.2.6 RADIUS OF PROXIMITY

The user is given the ability to control point label radius of proximity so as to optimize LABPOS to cope with regions of different label density (Rule [6.7]). For instance it was envisaged that by reducing the radius of proximity, labels are moved closer in towards their points and consequently give more room for placement and less conflicts. Fig 6.30 certainly confirms that the number of positions increases slightly with a decrease in radius of

proximity, but contrary to the above view Fig 6.31 shows no obvious trends in label conflicts for grid squares of high label density and for the least dense grid square the reverse is the case. One possible reason for the latter is that when the radius of proximity is increased, point label loci are increased in length and consequently the labels have more dispersed positions available and thus a better chance to move out of regions of conflict. This does not apply to the more label dense grid squares where the movement of labels away from one region of conflict may move them into another.

Despite these problems, a label radius of proximity of 1.1 character block widths has been selected since in all three grid squares the number of label conflicts is near a local minimum in the curves. Figs 6.32, 6.33 and 6.34 show the effects on the map appearance of increasing the point label radius of proximity.

6.9.2.7 LABEL SEPARATION OR BUFFER DISTANCE

One of the disadvantages of measuring the performance of name placement by the number of label conflicts is that a "label conflict" includes the buffer distance (Rule [6.13]) and so if this is increased, the number of label conflicts increases (Figs 6.35 and 6.40). In all the optimization efforts so far the buffer or separation

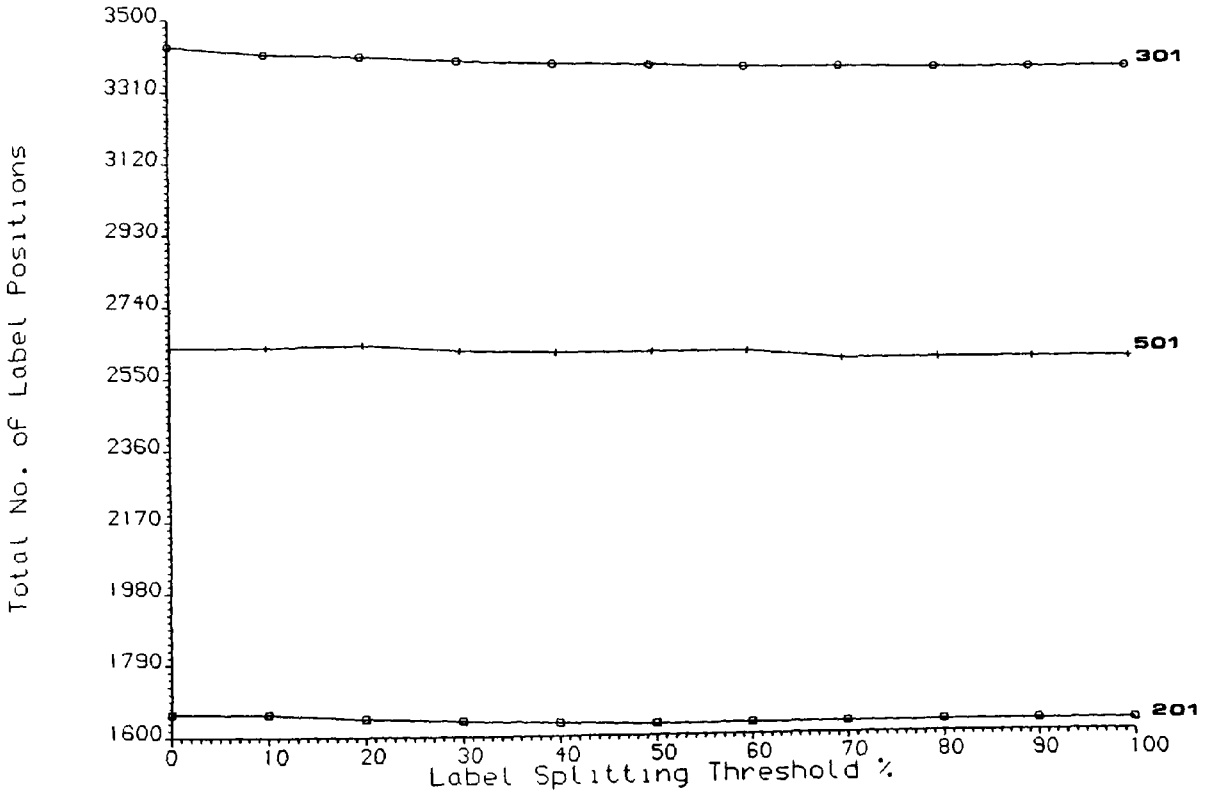
distance has been set to zero so that label conflicts correspond to label overlaps. For the purposes of optimizing the label buffer or separation distance, label overlap will be used as a measure of quality. Because of the pronounced horizontal alignment of most labels, the X buffer distance will be investigated prior to the Y buffer distance.

The curves presented in Fig 6.36 and the maps presented in Fig 6.37, 6.38 and 6.39 show that there is a slight increase in the number of label overlaps with an increase in the X buffer. Therefore a value was selected from Table 6.11 where a minimum number of overlaps occurred in all three library squares and which also allows sufficient separation between labels. An X buffer or separation distance of 0.8 character widths was selected. This was then used to generate the set of data for the Y buffer for which a trend of increasing conflict with buffer distance can be seen in Fig 6.41 and Fig 6.42, 6.43 and 6.44. Clearly the Y buffer distance has to be set small but not at the expense of ambiguity, therefore a value of 0.6 character widths was selected.

Table 6.9 Statistics for label splitting threshold optimization.

Label Splitting Threshold	Library Grid Square 201			Library Grid Sqaure 301			Library Grid Square 501		
	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions
0%	1661	4	6	3429	32	10	2632	16	16
10%	1660	4	6	3408	32	15	2633	17	10
20%	1648	4	6	3402	30	15	2639	16	10
30%	1641	4	6	3390	30	15	2624	18	10
40%	1635	7	6	3382	31	15	2618	18	8
50%	1630	7	6	3378	31	15	2618	18	8
60%	1630	7	6	3372	31	15	2618	18	8
70%	1630	7	6	3372	31	15	2596	18	8
80%	1630	7	6	3368	31	15	2596	18	8
90%	1630	7	6	3368	31	15	2596	18	8
100%	1630	7	6	3368	31	15	2596	18	8

Fig 6.27 Graph of total number of positions available to labels versus label splitting threshold.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.28 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Label splitting threshold 20%.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	60.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.29 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Label splitting threshold 60%.

Table 6.10 Statistics for radius of proximity optimization.

Radius of Proximity (Char widths)	Library Grid Square 201			Library Grid Sqaure 301			Library Grid Square 501		
	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions	Posi-tions	Confl-icts	Itera-tions
0.3	1672	12	6	3541	39	10	2727	26	9
0.5	1674	9	8	3506	29	11	2705	27	9
0.7	1670	6	8	3451	31	15	2670	29	9
0.9	1653	6	6	3445	39	15	2645	28	9
1.1	1660	4	6	3402	30	15	2639	16	10
1.3	1639	0	2	3368	36	9	2613	27	10
1.5	1631	0	2	3346	43	11	2573	28	10
1.7	1613	2	3	3287	35	11	2577	25	10
1.9	1603	0	6	3243	40	16	2563	16	13

Fig 6.30 Graph of total number of positions available to labels versus radius of proximity.

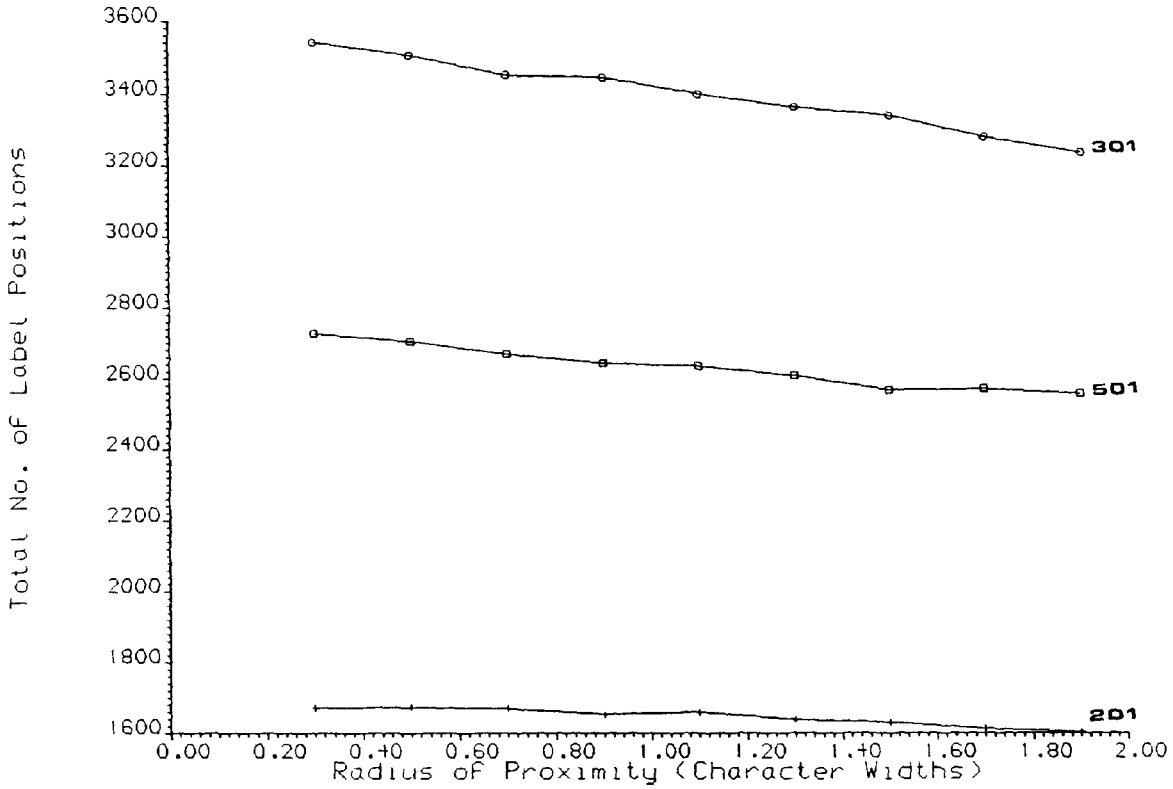
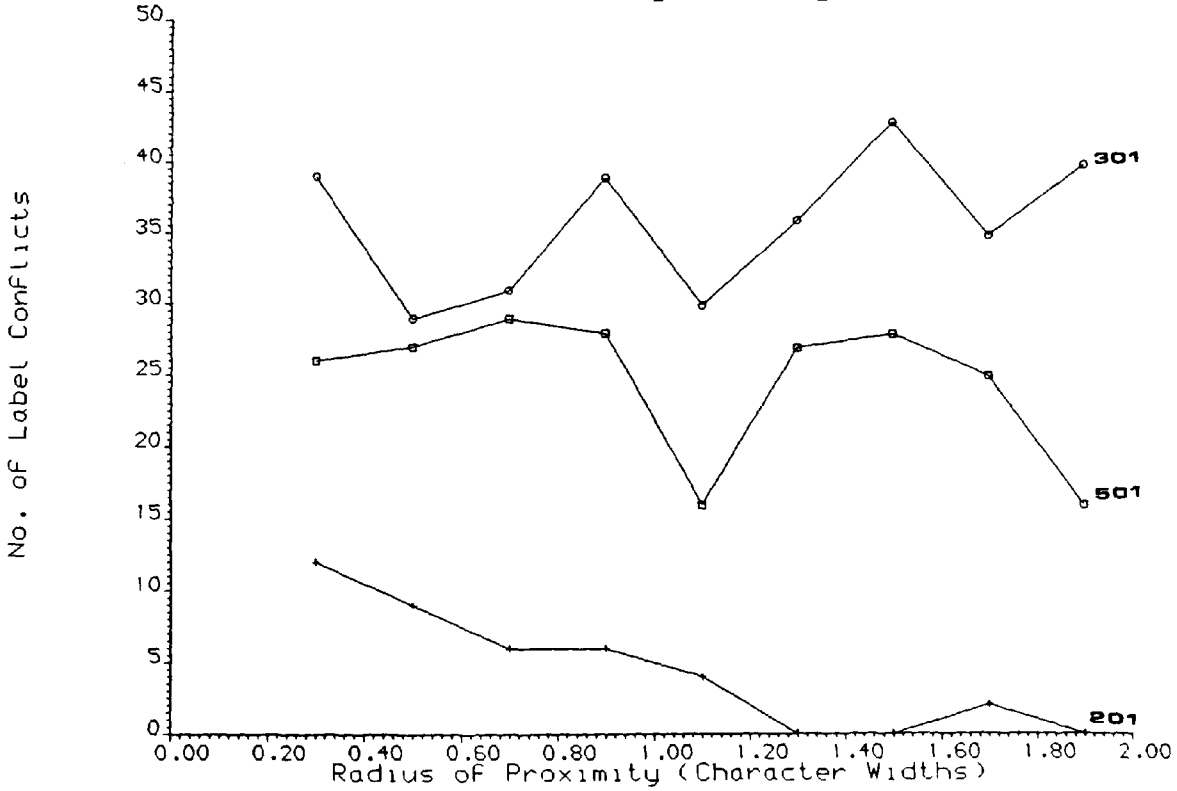


Fig 6.31 Graph of total number of label conflicts versus radius of proximity.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	0.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.32 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Radius of proximity: 0.5.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.33 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Radius of proximity: 1.1.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.5
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.34 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Radius of proximity: 1.5.

Table 6.11 Statistics for X-buffer or separation distance optimization.

X-Buffer Distance (Char widths)	Library Grid Square 201			Library Grid Sqaure 301			Library Grid Square 501		
	Confl-icts	Over-laps	Itera-tions	Confl-icts	Over-laps	Itera-tions	Confl-icts	Over-laps	Itera-tions
0.0	4	4	6	30	30	15	16	16	10
0.2	4	4	6	31	31	15	23	22	10
0.4	3	3	7	33	32	11	23	22	10
0.6	4	4	6	33	33	11	27	24	10
0.8	4	4	6	32	32	12	29	26	10
1.0	4	4	6	33	33	11	34	28	13
1.2	5	4	6	35	33	11	36	28	13
1.4	6	4	6	40	36	15	37	28	13
1.6	8	4	6	44	36	11	36	26	12
1.8	8	4	6	44	34	11	37	26	12
2.0	8	4	6	46	34	12	39	26	12

Table 6.12 Statistics for y-buffer or separation distance optimization.

Y-Buffer Distance (Char widths)	Library Grid Square 201			Library Grid Sqaure 301			Library Grid Square 501		
	Confl-icts	Over-laps	Itera-tions	Confl-icts	Over-laps	Itera-tions	Confl-icts	Over-laps	Itera-tions
0.0	4	4	6	32	32	12	29	26	10
0.2	8	4	6	46	33	11	39	19	10
0.4	17	4	8	67	36	16	44	21	10
0.6	18	4	7	77	38	10	54	21	10
0.8	25	4	7	96	41	12	69	23	10
1.0	33	4	8	118	45	15	81	23	12
1.2	42	4	9	137	47	10	94	25	11
1.4	55	6	10	150	42	12	111	25	12
1.6	65	6	10	179	48	13	130	29	12

Fig 6.35 Graph of total number of label conflicts versus X-buffer or separation distance.

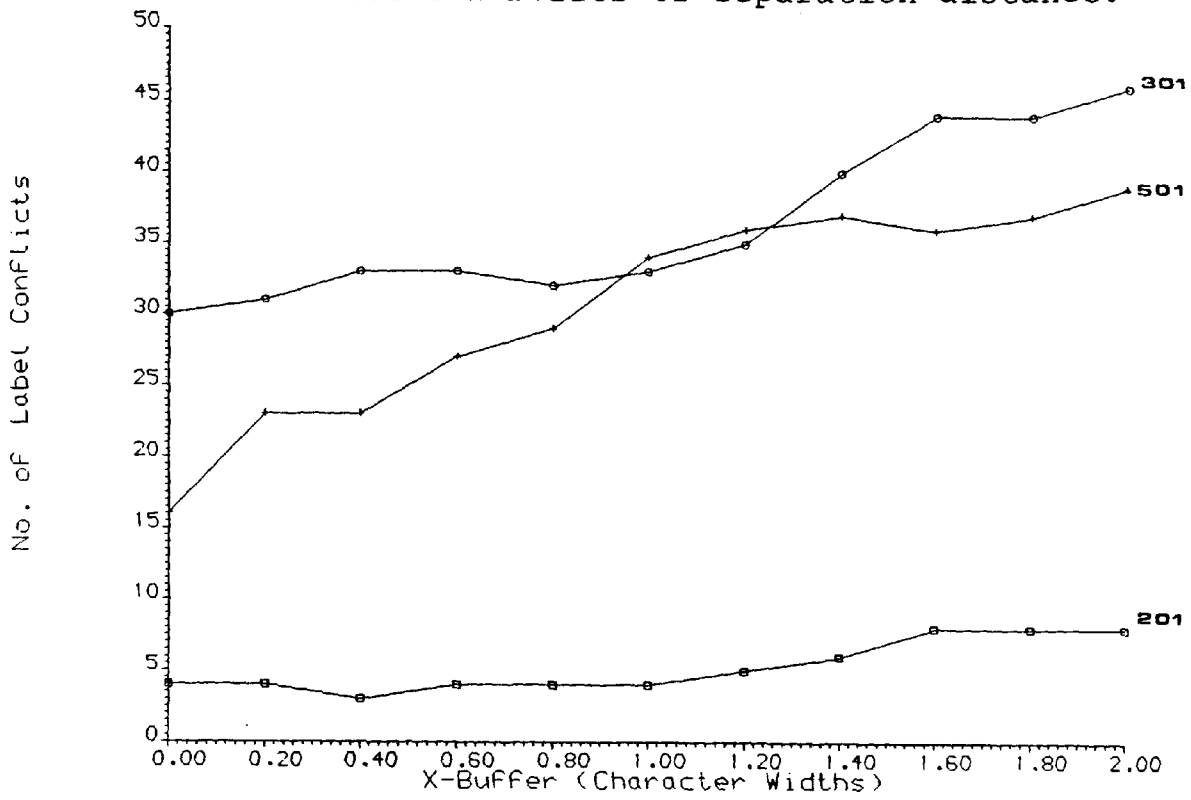
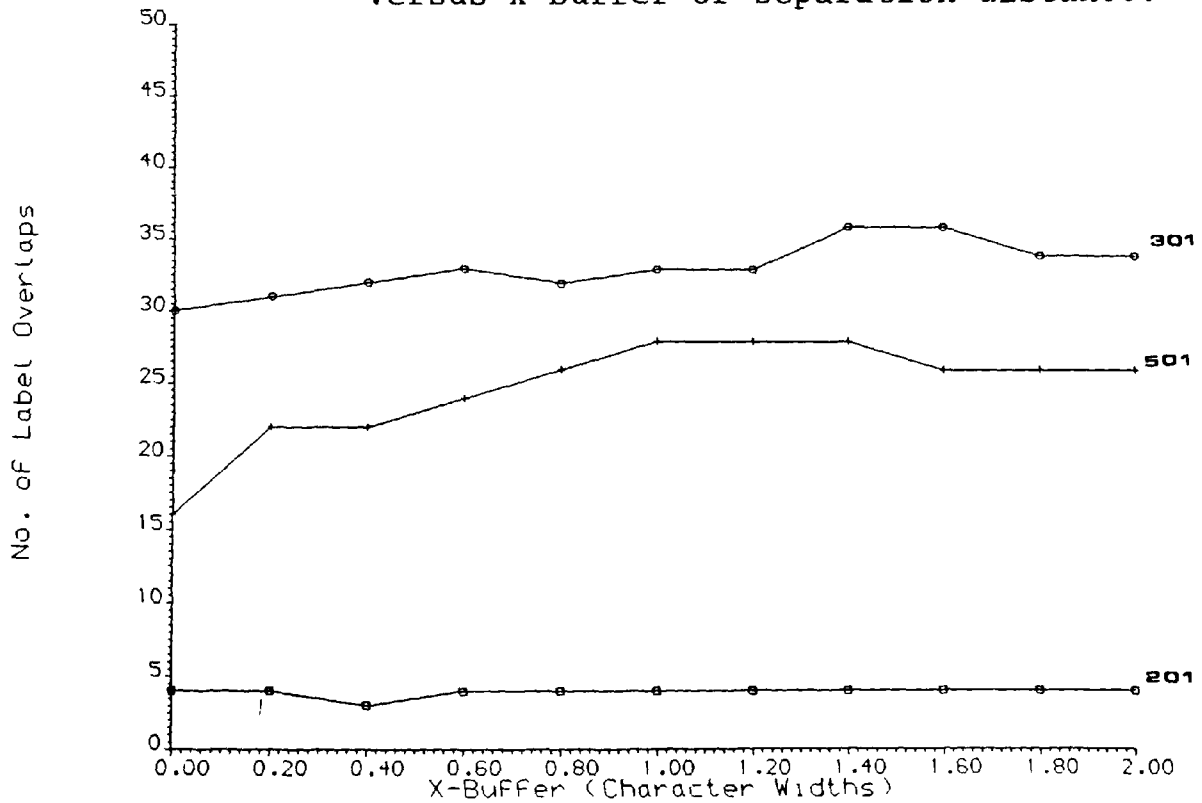
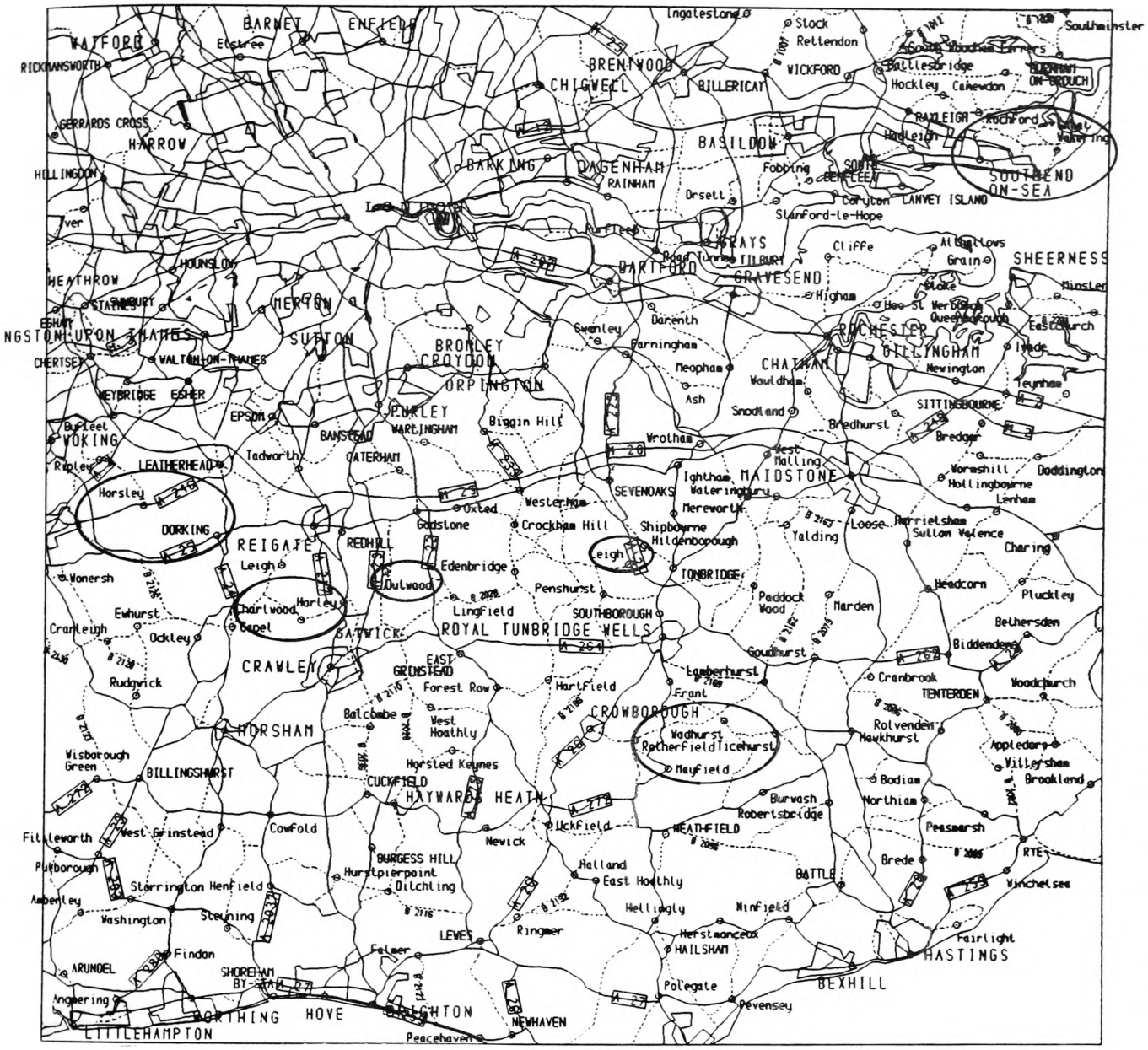


Fig 6.36 Graph of total number of label overlaps versus X-buffer or separation distance.

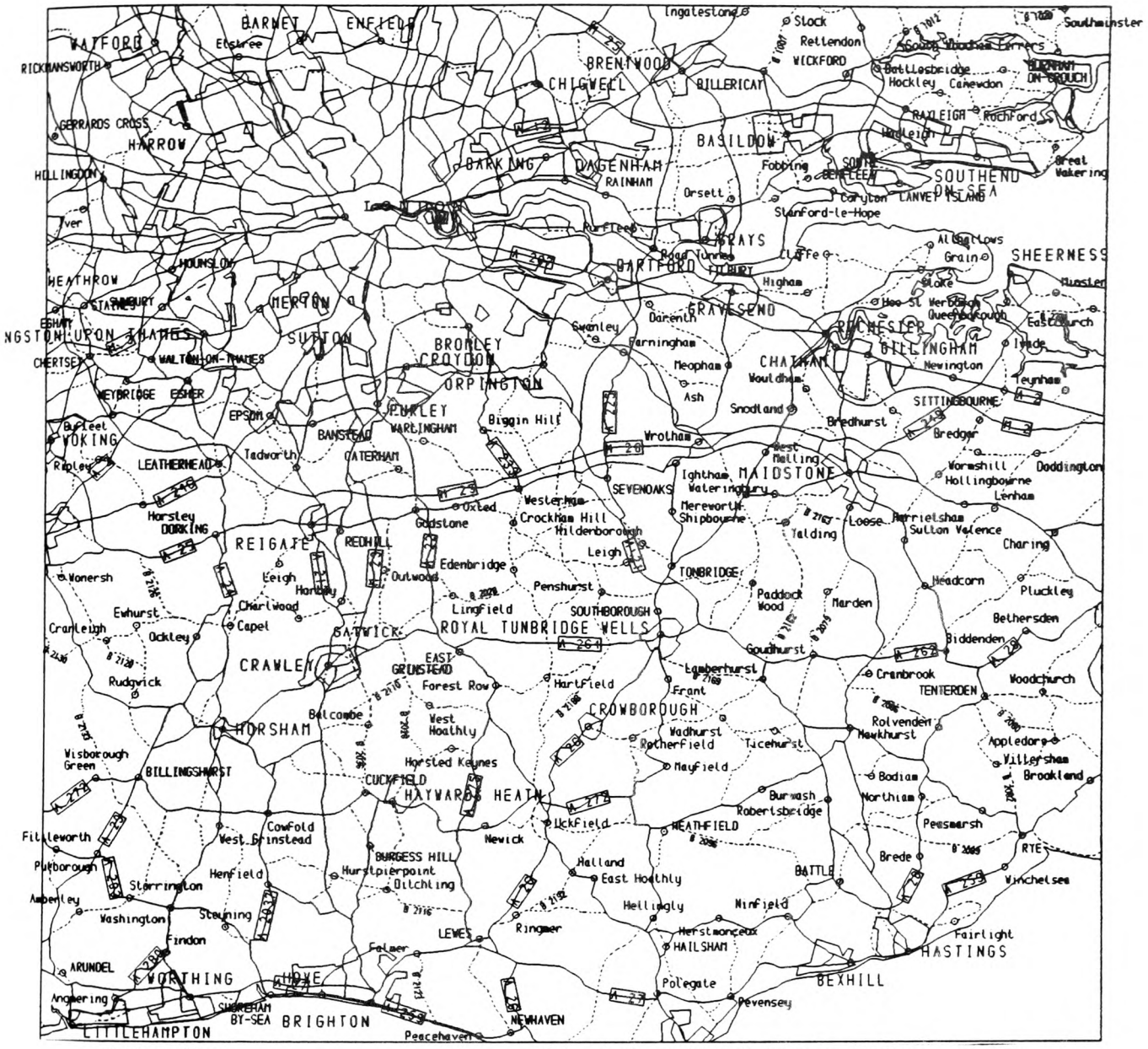




Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.0
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.37 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Label X-buffer: 0.0.


BARNET
ENFIELD
WATFORD
Elstree
RICKMANSWORTH
GERRARDS CROSS
HARROW
HILLINGDON
Iver
HEATHROW
STAINES
EGHAM
CHERTSEY
WALTON-ON-THAMES
WEYBRIDGE
ESHER
Byfleet
WOKING
Ripley
LEATHERHEAD
Horsley
DORKING
Wonersh
Ewhurst
Cranleigh
Ockley
Rudgwick
CRAWLEY
Wisborough Green
BILLINGSHURST
Fittleworth
Pulborough
Amberley
Washington
ARUNDEL
Angmering
LITTLEHAMPTON
WORTHING
SHOREHAM BY SEA
HOVE
BRIGHTON
Peacehaven
NEWHAVEN
LEWES
Ringmer
Falmer
Findon
Steyning
Henfield
Storrington
West Grinstead
Cowfold
BURGESS HILL
Hurstpierpoint
Ditchling
Newick
HAYWARDS HEATH
CUCKFIELD
Horsted Keynes
HORSHAM
Balcombe
West Hoathly
EAST GRINSTEAD
Forest Row
CROWBOROUGH
Hartfield
Frant
Wadhurst
Ticehurst
Rotherfield
Mayfield
Uckfield
Halland
East Hoathly
Hellingly
Herstmonceux
HAILSHAM
Polegate
Pevensey
BEXHILL
BATTLE
Robertsbridge
Burwash
HEATHFIELD
Mountfield
Brede
Peasmarsh
RYE
Winchelsea
Fairlight
HASTINGS
Rolvenden
Hawkhurst
Cranbrook
TENTERDEN
Biddenden
Bethersden
Appledore
Woodchurch
Brookland
Ashford
Headcorn
Pluckley
Charing
Harrietsham
Sutton Valence
Loose
Yalding
Paddock Wood
Marden
Goudhurst
Lamberhurst
TONBRIDGE
Penshurst
SOUTHBOROUGH
ROYAL TUNBRIDGE WELLS
Lingfield
Edenbridge
Leigh
Oxted
Godstone
Crockham Hill
Westerham
Shipbourne
Hildenborough
Tonbridge
SEVENOAKS
Ightham
Wateringbury
Mereworth
MAIDSTONE
West Malling
Vanwshill
Hollingbourne
Lenham
Doddington
Teynham
SITTINGBOURNE
Bredhurst
Bredgar
Snodland
Wrotham
Ash
Meopham
Wouldham
CHATHAM
Newington
RAINHAM
Swanley
Farningham
Darenth
Dartford
Purfleet
GRAYS
TILBURY
GRAVESEND
Higham
Cliffe
Grain
Stoke
Allhallows
SHEERNESS
Minster
Queenborough
Eastchurch
Leysdown
Warden Point
Warden Bay
Harty Ferry
Elmley
Sheppey
Isle of Sheppey
Sittingbourne
Faversham
Whitstable
Herne Bay
Reculver
Margate
Ramsgate
Broadstairs
Deal
Sandwich
Ramsgate
Minster
Ebbsfleet
Northfleet
Gravesend
Tilbury
Chadwell St Mary
Grays
Purfleet
Dartford
Crayford
Bexley
Erith
Thamesmead
Woolwich
Plumstead
Abbey Wood
Belvedere
Bexleyheath
Sidcup
Chislehurst
Orpington
Petts Wood
Bromley
Croydon
Beckenham
Penge
Anerley
Crystal Palace
Sydenham
Forest Hill
Lewisham
Deptford
Greenwich
Blackheath
Eltham
Mottingham
New Eltham
Kidbrooke
Charlton
Woolwich
Barking
Dagenham
Rainham
Romford
Hornchurch
Upminster
Cranham
Hornchurch
Elm Park
Dagenham
Barking
Ilford
Gants Hill
Redbridge
Wanstead
Leytonstone
Walthamstow
Chingford
Epping
Loughton
Buckhurst Hill
Woodford
South Woodford
Snaresbrook
Leyton
Stratford
Bow
Mile End
Whitechapel
Aldgate East
Tower Hill
Fenchurch Street
Liverpool Street
Bishopsgate
Shoreditch
Hoxton
Dalston
Hackney
Stoke Newington
Tottenham
Edmonton
Enfield
Southgate
Barnet
Finchley
Hendon
Golders Green
Hampstead
Kilburn
Willesden
Cricklewood
Neasden
Dollis Hill
Wembley
Kingsbury
Queensbury
Stanmore
Edgware
Mill Hill
Totteridge
Whetstone
North Finchley
Friern Barnet
Muswell Hill
Alexandra Palace
Wood Green
Tottenham
Seven Sisters
Finsbury Park
Highbury
Islington
Angel
Clerkenwell
Farringdon
Holborn
Bloomsbury
Euston
Camden Town
Kentish Town
Gospel Oak
Hampstead Heath
Highgate
Crouch End
Hornsey
Finsbury Park
Stoke Newington
Dalston
Hackney
Shoreditch
Bethnal Green
Whitechapel
Aldgate East
Tower Hill
Fenchurch Street
Liverpool Street
Bishopsgate
Moorgate
Barbican
Farringdon
Holborn
Bloomsbury
Euston
Camden Town
Kentish Town
Gospel Oak
Hampstead Heath
Highgate
Crouch End
Hornsey
Wood Green
Alexandra Palace
Muswell Hill
Friern Barnet
North Finchley
Whetstone
Totteridge
Mill Hill
Edgware
Stanmore
Queensbury
Kingsbury
Neasden
Dollis Hill
Cricklewood
Willesden
Kilburn
Hampstead
Golders Green
Hendon
Finchley
Barnet
Southgate
Enfield
Edmonton
Tottenham
Seven Sisters
Finsbury Park
Highbury
Islington
Angel
Clerkenwell



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	1.6
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.39 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Label X-buffer: 1.6.

Fig 6.40 Graph of total number of label conflicts versus Y-buffer or separation distance.

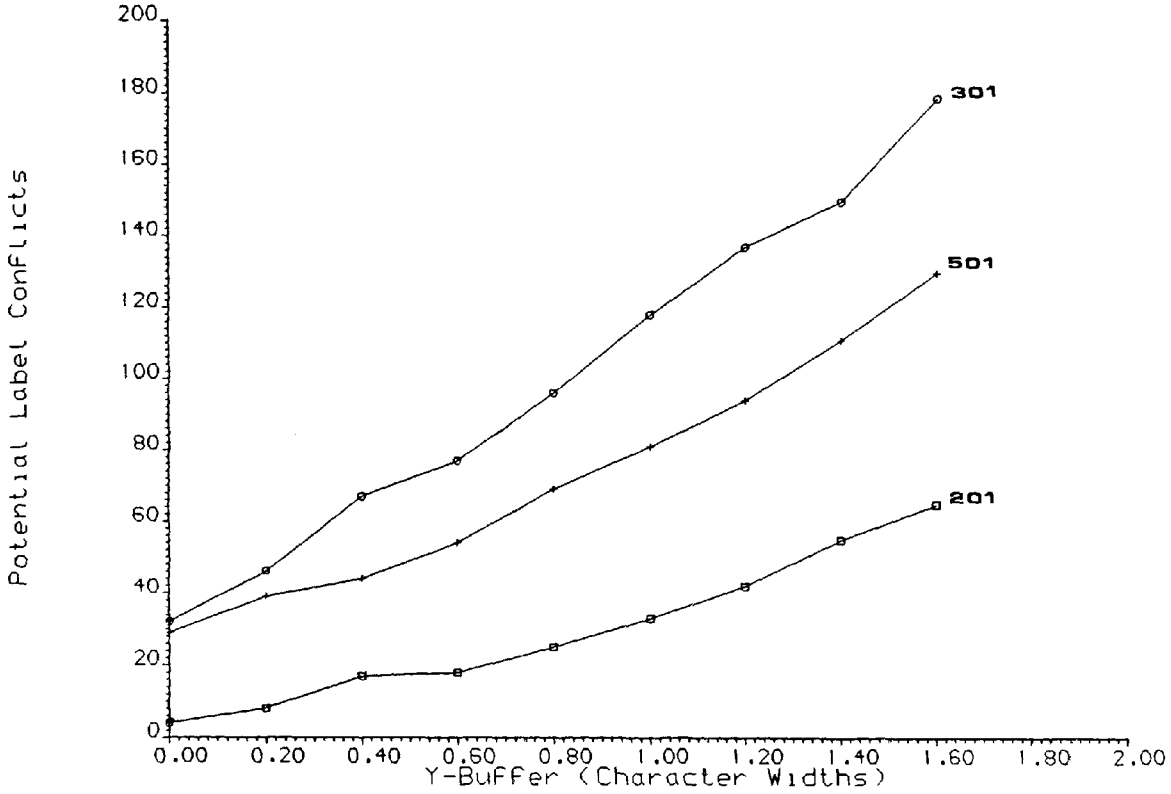
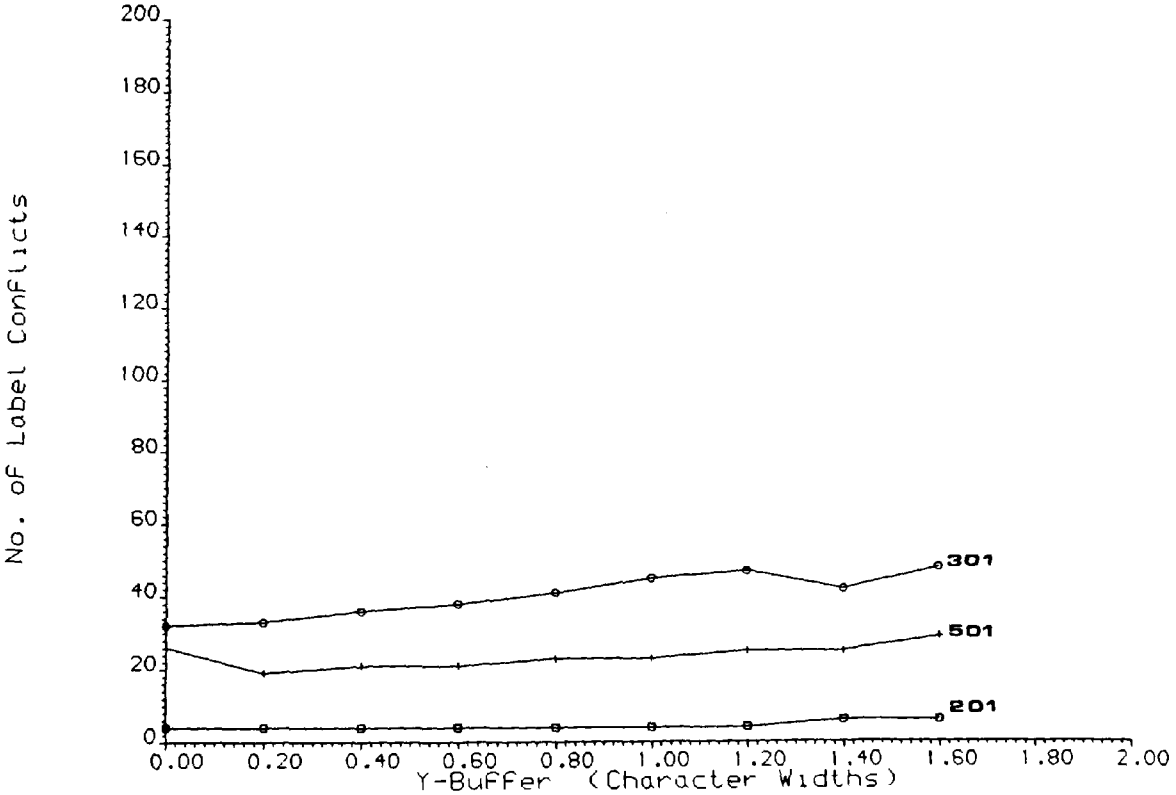


Fig 6.41 Graph of total number of label overlaps versus Y-buffer or separation distance.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.0
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.42 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Label Y-buffer: 0.0.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Dense space threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.43 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Label Y-buffer: 0.6.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	1.2
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

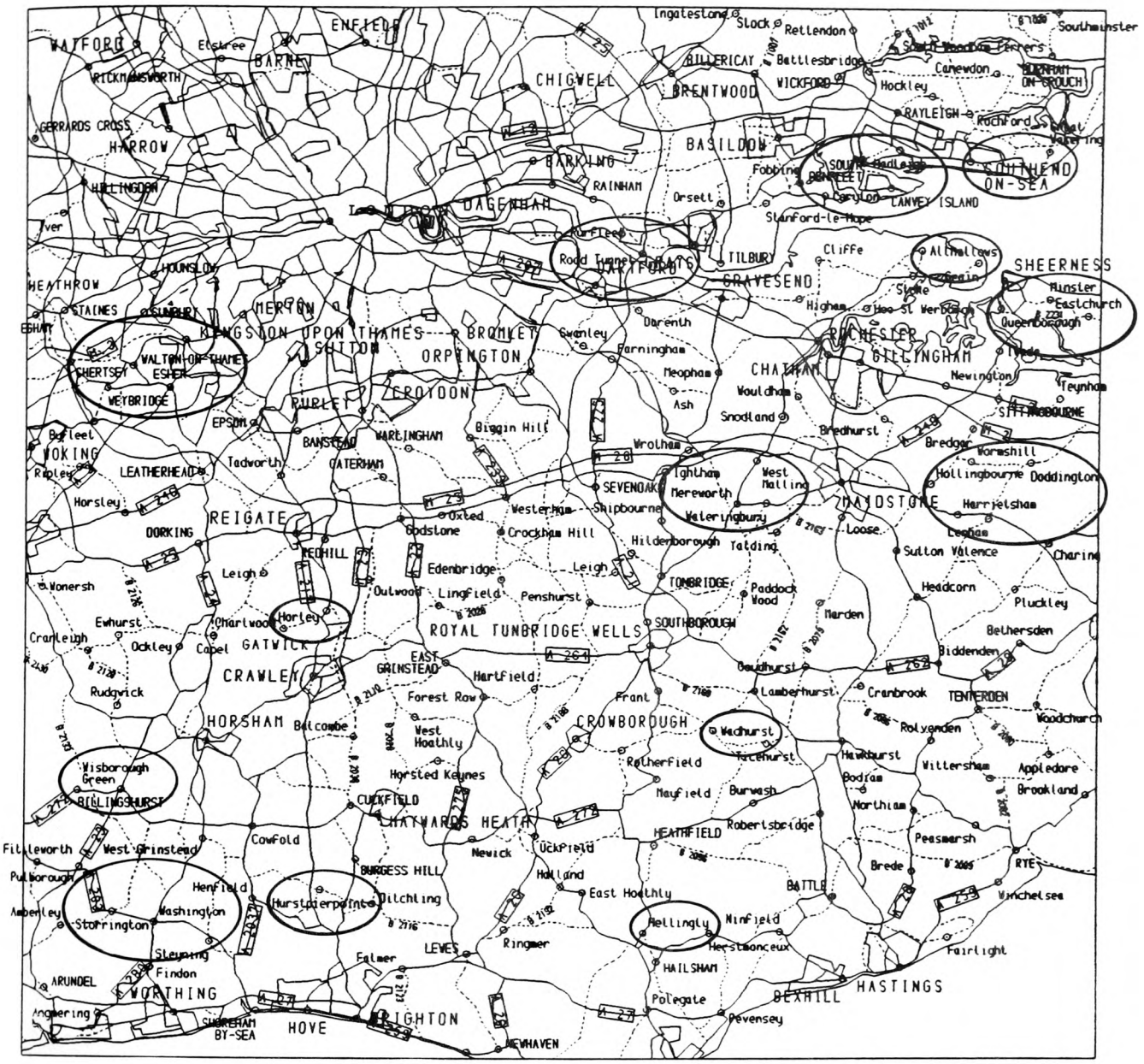
Fig 6.44 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Label Y-buffer: 1.2.

6.9.2.8 SETTLEMENT SYMBOL WIDTH

When optimizing settlement symbol width in the raster image it is best to judge the results by the amount of ambiguity present on the map produced. The symbol width should be bigger than the radius of proximity for the feature concerned in order to avoid ambiguity. Note that changing the feature width in the raster image is not represented by a change in size on the plotted map, it just alters the concept of symbol width size to the program. Figs 6.45 to 6.48 illustrate that increasing settlement symbol width decreases ambiguity but appears to result in an increase in the number of label overlaps. It was decided to carry on using the original default settlement symbol widths (Table 6.5).

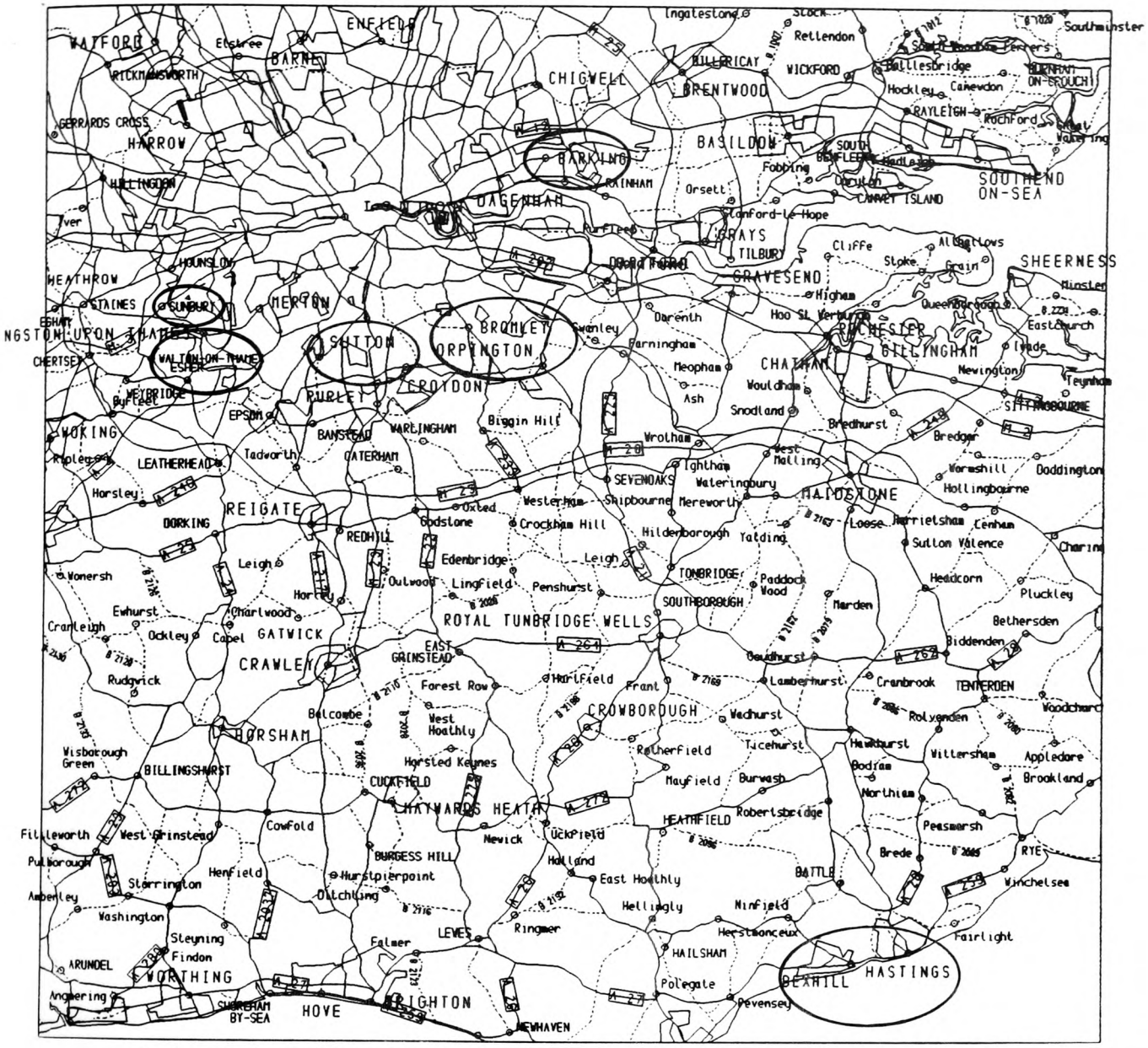
6.9.2.9 APPLICATION OF THE OPTIMIZED RULES AND SPECIFICATIONS

Fig 6.49, 6.50, and 6.51 show the effects of the optimization on the three example grid squares. Clearly problems remain, for instance in Fig 6.49, "Lynmouth" near "LYNTON" suffers from close proximity to nearby settlement symbols. Because of this all its available placements overlie at least one high priority settlement pixel and as a result all positions are illegal and so it is placed in position 1. To resolve this situation, both the radius of



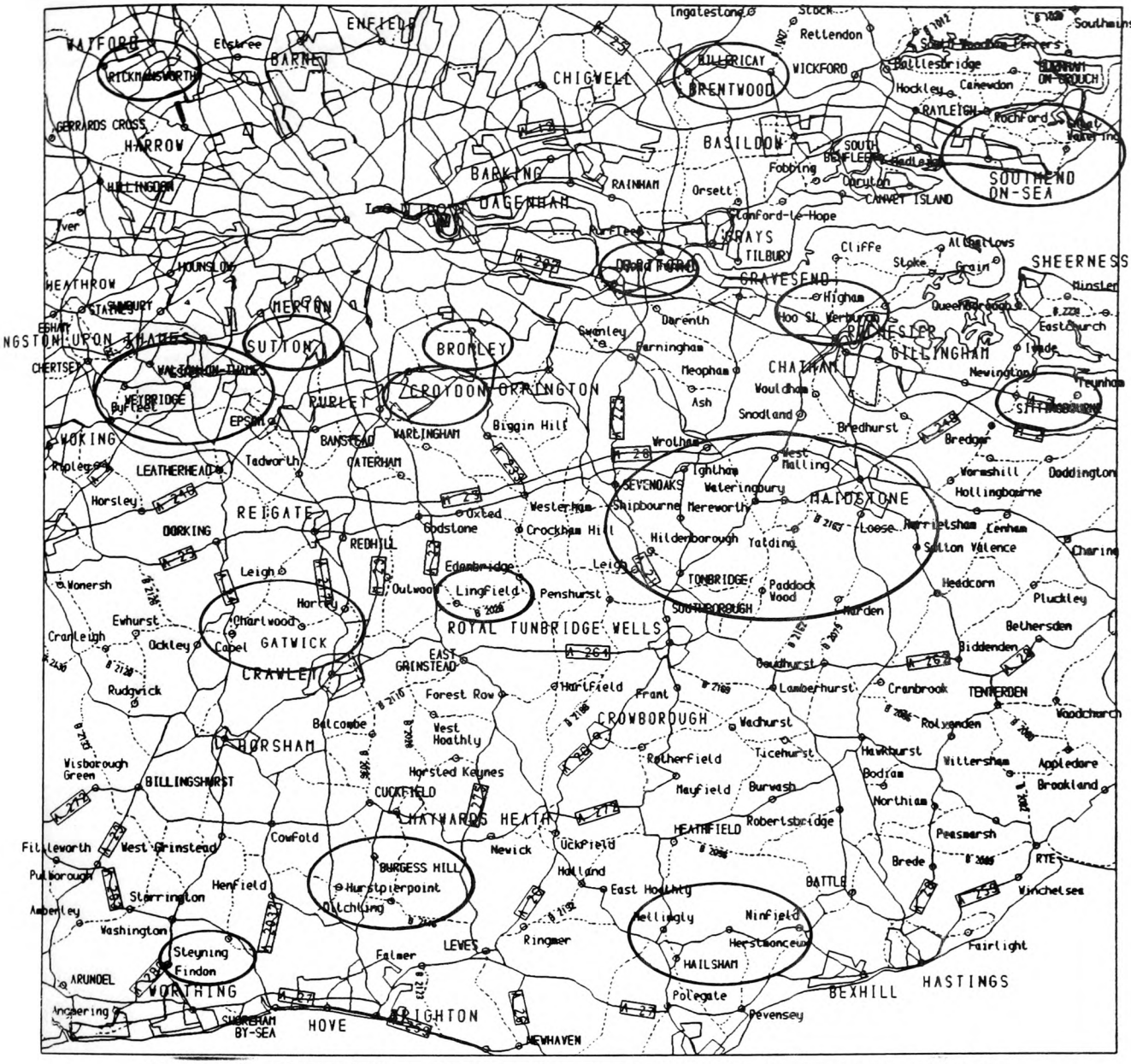
Feat. Code	Token	Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n		1600	3	0	0	0	0	0	0
1	mway_service_ltd_e		1600	3	0	0	0	0	0	0
2	mway_service_ltd_s		1600	3	0	0	0	0	0	0
3	mway_service_ltd_w		1600	3	0	0	0	0	0	0
20	motorway		600	3	810	625	540	202	1313	900
21	motorway_junction		1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs		1600	3	0	0	0	0	0	0
23	mway_service_area		1600	3	0	0	0	0	0	0
30	primary_route_s_c		450	1	0	0	0	0	0	0
31	primary_route_d_c		550	2	810	625	540	202	1250	800
32	prim_route_narrow		400	1	810	625	540	202	1250	800
35	prim_route_srv_area		1600	1	0	0	0	0	0	0
40	main_road_s_c		300	1	810	625	540	202	1250	800
41	main_road_d_c		550	2	810	625	540	202	1250	800
42	main_road_narrow		300	1	810	625	540	202	1250	800
49	b_road		125	0	600	438	400	150	625	438
50	other_road		125	0	0	0	0	0	0	0
61	city		2000	4	1075	750	720	269	1375	1040
62	large_town		1200	4	1075	688	720	269	1312	938
63	village_on_prim_rt		600	4	850	600	500	188	1125	625
64	town_on_prim_route		600	4	1075	688	720	269	1312	888
65	small_town_village		600	4	850	600	500	188	1125	625
75	airport		600	1	900	700	500	188	1100	900

Fig 6.45 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Symbol feature width optimization.



Feat. Code	Token	Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n		1600	3	0	0	0	0	0	0
1	mway_service_ltd_e		1600	3	0	0	0	0	0	0
2	mway_service_ltd_s		1600	3	0	0	0	0	0	0
3	mway_service_ltd_w		1600	3	0	0	0	0	0	0
20	motorway		600	3	810	625	540	202	1313	900
21	motorway_junction		1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs		1600	3	0	0	0	0	0	0
23	mway_service_area		1600	3	0	0	0	0	0	0
30	primary_route_s_c		450	1	0	0	0	0	0	0
31	primary_route_d_c		550	2	810	625	540	202	1250	800
32	prim_route_narrow		400	1	810	625	540	202	1250	800
35	prim_route_srv_area		1600	1	0	0	0	0	0	0
40	main_road_s_c		300	1	810	625	540	202	1250	800
41	main_road_d_c		550	2	810	625	540	202	1250	800
42	main_road_narrow		300	1	810	625	540	202	1250	800
49	b_road		125	0	600	438	400	150	625	438
50	other_road		125	0	0	0	0	0	0	0
61	city		3000	4	1075	750	720	269	1375	1040
62	large_town		1800	4	1075	688	720	269	1312	938
63	village_on_prim_rt		900	4	850	600	500	188	1125	625
64	town_on_prim_route		900	4	1075	688	720	269	1312	888
65	small_town_village		900	4	850	600	500	188	1125	625
75	airport		900	1	900	700	500	188	1100	900

Fig 6.46 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Symbol feature width optimization.



Feat. Code	Token Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n	1600	3	0	0	0	0	0	0
1	mway_service_ltd_e	1600	3	0	0	0	0	0	0
2	mway_service_ltd_s	1600	3	0	0	0	0	0	0
3	mway_service_ltd_w	1600	3	0	0	0	0	0	0
20	motorway	600	3	810	625	540	202	1313	900
21	motorway_junction	1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs	1600	3	0	0	0	0	0	0
23	mway_service_area	1600	3	0	0	0	0	0	0
30	primary_route_s_c	450	1	0	0	0	0	0	0
31	primary_route_d_c	550	2	810	625	540	202	1250	800
32	prim_route_narrow	400	1	810	625	540	202	1250	800
35	prim_route_srv_area	1600	1	0	0	0	0	0	0
40	main_road_s_c	300	1	810	625	540	202	1250	800
41	main_road_d_c	550	2	810	625	540	202	1250	800
42	main_road_narrow	300	1	810	625	540	202	1250	800
49	b_road	125	0	600	438	400	150	625	438
50	other_road	125	0	0	0	0	0	0	0
61	city	4000	4	1075	750	720	269	1375	1040
62	large_town	2400	4	1075	688	720	269	1312	938
63	village_on_prim_rt	1200	4	850	600	500	188	1125	625
64	town_on_prim_route	1200	4	1075	688	720	269	1312	888
65	small_town_village	1200	4	850	600	500	188	1125	625
75	airport	1200	1	900	700	500	188	1100	900

Fig 6.47 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Symbol feature width optimization.

Feat. Code	Token Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n	1600	3	0	0	0	0	0	0
1	mway_service_ltd_e	1600	3	0	0	0	0	0	0
2	mway_service_ltd_s	1600	3	0	0	0	0	0	0
3	mway_service_ltd_w	1600	3	0	0	0	0	0	0
20	motorway	600	3	810	625	540	202	1313	900
21	motorway_junction	1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs	1600	3	0	0	0	0	0	0
23	mway_service_area	1600	3	0	0	0	0	0	0
30	primary_route_s_c	450	1	0	0	0	0	0	0
31	primary_route_d_c	550	2	810	625	540	202	1250	800
32	prim_route_narrow	400	1	810	625	540	202	1250	800
35	prim_route_srv_area	1600	1	0	0	0	0	0	0
40	main_road_s_c	300	1	810	625	540	202	1250	800
41	main_road_d_c	550	2	810	625	540	202	1250	800
42	main_road_narrow	300	1	810	625	540	202	1250	800
49	b_road	125	0	600	438	400	150	625	438
50	other_road	125	0	0	0	0	0	0	0
61	city	5000	4	1075	750	720	269	1375	1040
62	large_town	3000	4	1075	688	720	269	1312	938
63	village_on_prim_rt	1500	4	850	600	500	188	1125	625
64	town_on_prim_route	1500	4	1075	688	720	269	1312	888
65	small_town_village	1500	4	850	600	500	188	1125	625
75	airport	1450	1	900	700	500	188	1100	900

Fig 6.48 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Symbol feature width optimization.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.49 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Produced using selected optimized rules/specifications.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.50 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Produced using selected optimized rules/specifications.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.51 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Produced using selected optimized rules/specifications.

proximity for point labels and point symbol widths could be reduced, however this would be at the expense of clarity elsewhere on the map.

Figs 6.50 and 6.51 are reasonably satisfactory except that city labels often clash with other smaller settlement labels. For example in Fig 6.51 "SUNBURY" overlaps "STAINES" due to its close proximity to the "KINGSTON UPON THAMES" symbol. This could be resolved by reducing symbol widths for cities or reducing city label size. Another problem is the presence of a small number of overlaps between road and settlement labels for instance "HAYWARDS HEATH" and "A275" in Fig 6.51. These could be avoided by the use of horizontal line labels or the ability to label different sections of line.

6.9.3 MAP DESIGN

The user has control over several aspects of map design. For instance labels can be prevented from splitting (Fig 6.52) by simply setting a flag in rule [6.8]. If the user requires a map without labels being placed over the edge of the map, then this can easily be arranged by setting the "over edge" priority to that of settlements (Rule [6.2]). However this is not advisable since it usually leads to more label conflicts (Fig 6.53).

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	T
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.52 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - No label splitting allowed.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	999.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.53 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - No labels allowed over map border.

If it is desired to leave certain features as free as possible from overlying labels then the priorities of such features can be increased (Rule [6.2]). This has been illustrated in Fig 6.54 and Fig 6.55 where the priority of "Motorways" and "A class roads" respectively has been increased by a factor of ten times. Although this achieves the desired effect, it generally causes label conflicts elsewhere.

Selecting which lines to label is achieved by only labelling lines that are long enough to take the label length plus a buffer distance on each end of the label (Rule [6.5]). Figs 6.56 to 6.59 illustrate that the effect of increasing the line buffer (Section 6.4.3) is that only the longer roads are labelled and consequently fewer road labels are selected. One advantage of this is that the number of conflicts is reduced due to a lower label density.

The preferred placement order of labels with respect to features can be altered (Rule [6.10] and [6.11]). For instance Fig 6.60 favours label positions at the start of lines rather than middle. In practice though the best positions with respect to underlying detail are found near the centres of the lines away from junctions.

It is also possible to make labels bigger or smaller by adjusting the label block characteristics (Rule [6.6]).

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	25.0	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.54 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - Motorway priority ten times greater.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	10.0	4th preferred line position	10
Bit plane 3 priority: Main routes	15.0	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

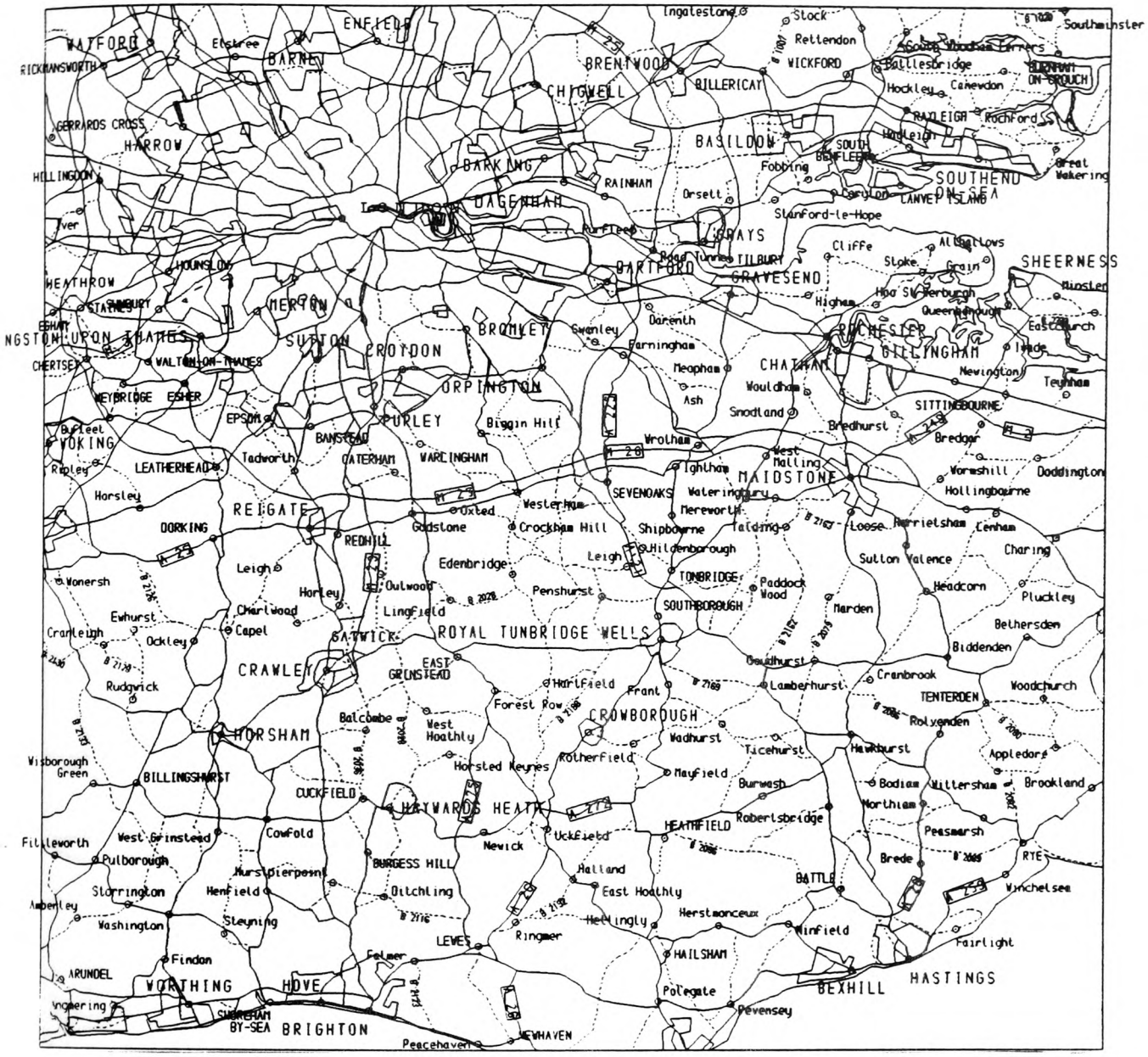
Fig 6.55 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - A class and main road priorities ten times greater.

Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	0.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.56 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Line buffer: 0.0.

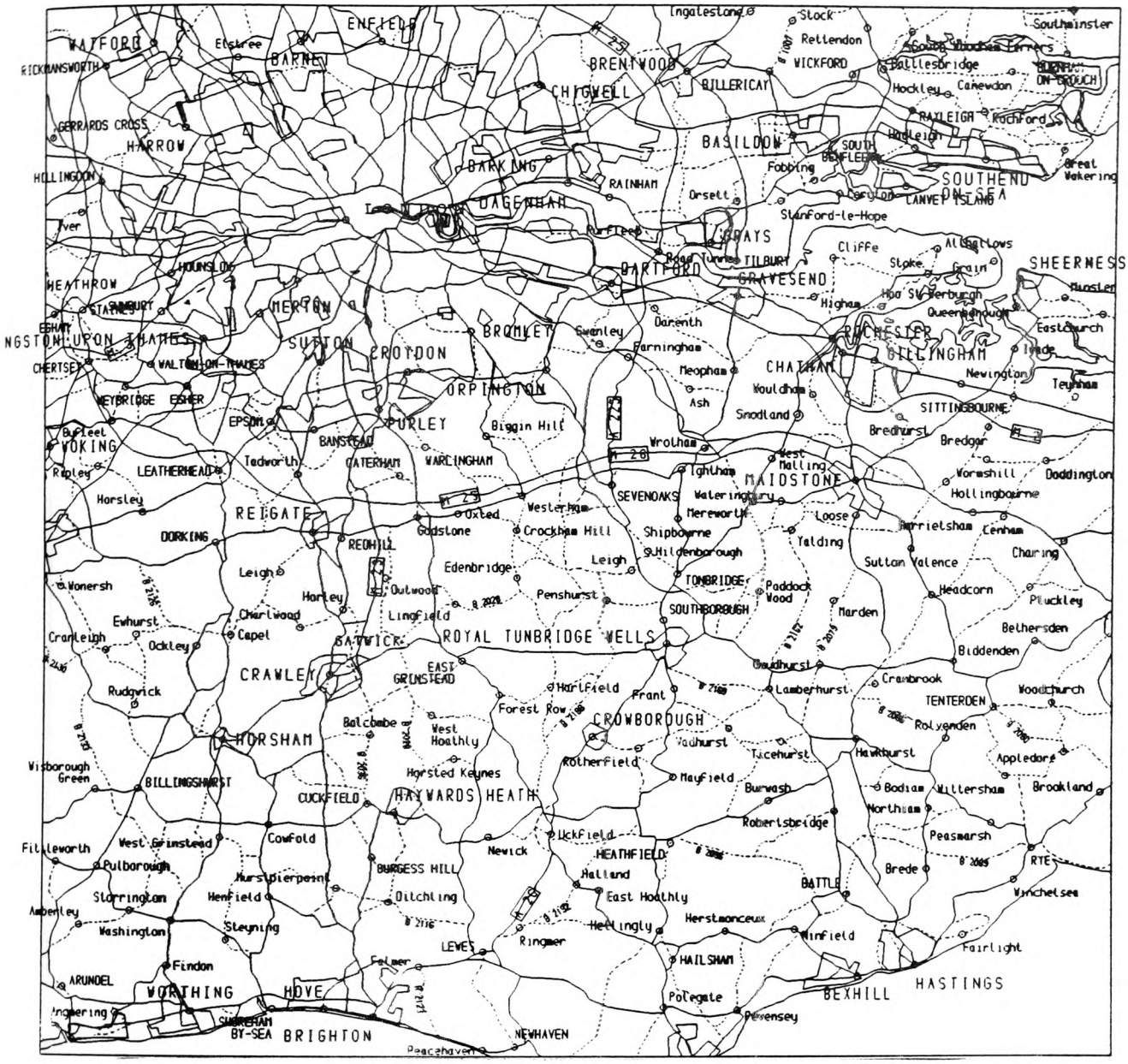
Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	1.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

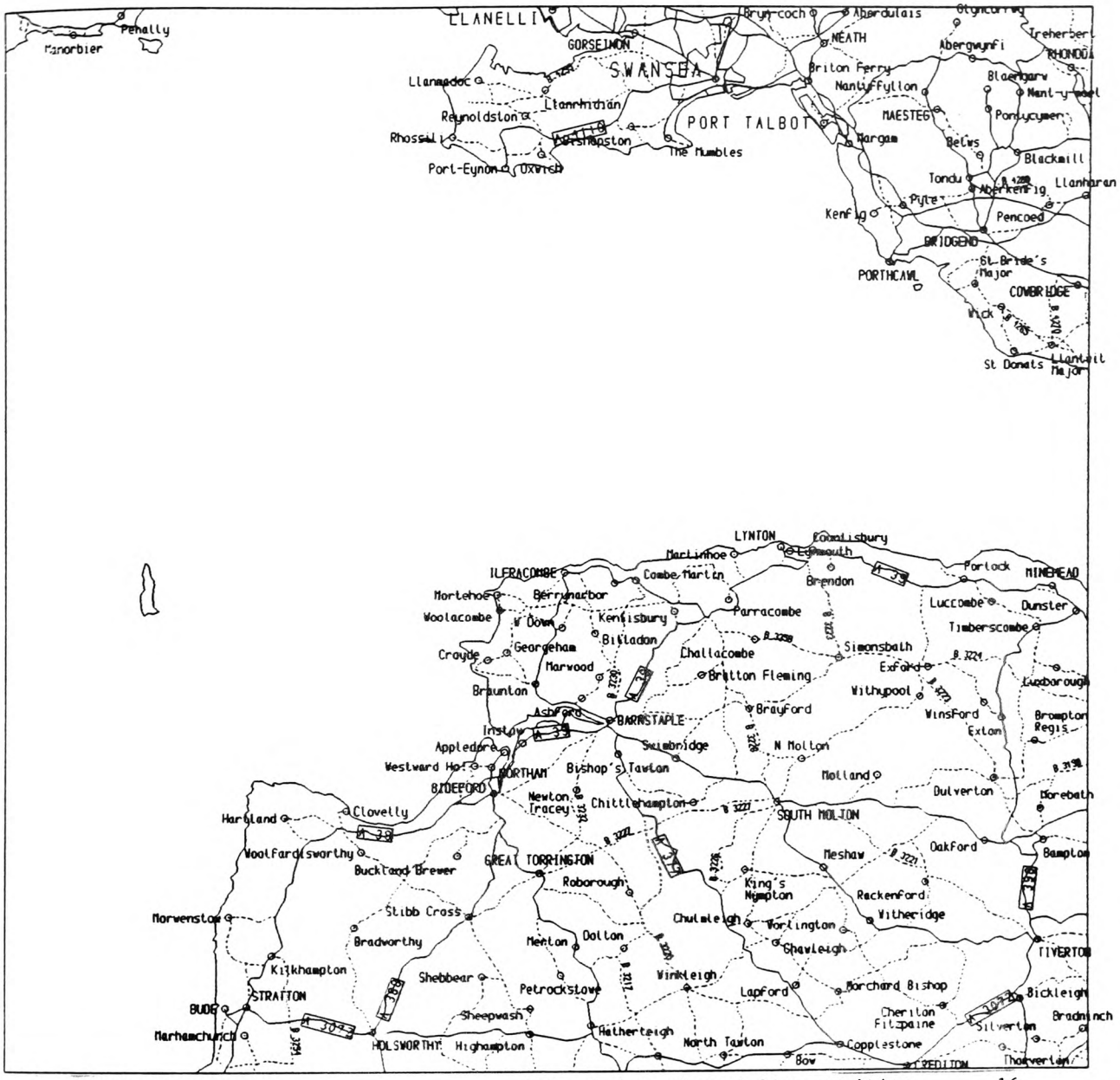
Fig 6.57 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Line buffer: 1.0.



Raster Size	350	2nd preferred line position	9
Bit plane 1 priority: B roads	0.3	3rd preferred line position	7
Bit plane 2 priority: A roads	1.0	4th preferred line position	10
Bit plane 3 priority: Main routes	1.5	5th preferred line position	6
Bit plane 4 priority: Motorways	2.5	6th preferred line position	11
Bit Plane 5 priority: Settlements	999.0	7th preferred line position	5
Over edge priority	1.0	8th preferred line position	12
Line label filter	T	9th preferred line position	4
Line label buffer zone	2.0	10th preferred line position	13
1st preferred point position	1	11th preferred line position	3
2nd preferred point position	13	12th preferred line position	14
3rd preferred point position	9	13th preferred line position	2
4th preferred point position	14	14th preferred line position	15
5th preferred point position	16	15th preferred line position	1
6th preferred point position	10	16th preferred line position	16
7th preferred point position	12	Label buffer (character widths)	T
8th preferred point position	2	Label vertical buffer	0.6
9th preferred point position	8	Label horizontal buffer	0.8
10th preferred point position	15	Radius of proximity (char. widths)	T
11th preferred point position	11	Radius of proximity	1.1
12th preferred point position	3	Densespace threshold % (x10)	500
13th preferred point position	7	Label splitting threshold %	20.0
14th preferred point position	5	Label not split	F
15th preferred point position	4	Label justification	3
16th preferred point position	6	Attempts before weighting label	2
1st preferred line position	8	Attempts before fixing label	5

Fig 6.58 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Line buffer: 2.0.


WATFORD
ENFIELD
BARNET
BRENTWOOD
CHIGWELL
BILLERICAY
WICKFORD
Rettendon
Stock
Ingatestone
Southminster
Hockley
Canewdon
RAYLEIGH
Rochford
HARROW
GERRARDS CROSS
HOLLINGDON
BARKING
RAINHAM
BASILDON
SOUTHEND ON SEA
Great Wakering
Fobbing
Orsett
Corringham
CANVEY ISLAND
Stanford-le-Hope
DAGENHAM
GRAYS
TILBURY
Cliffe
Stoke
Grain
SHEERNESS
HEATHROW
HOUNSLOW
DARTFORD
GRAVESEND
Higham
Hoo St Werburgh
Queenborough
Minster
STAINES
SUNBURY
MERTON
SUTTON
CROYDON
BROMLEY
Swanley
Farningham
Meopham
CHATHAM
RAINHAM
GILLINGHAM
Eastchurch
Newington
Teynham
Vouldham
Ash
Snodland
SITTINGBOURNE
Bredhurst
Bredgar
Wrotham
West Malling
MAIDSTONE
Vornashill
Doddington
Hollingbourne
Ightham
SEVENOAKS
Westerham
Oxted
Godstone
Crockham Hill
Shipbourne
Loose
Harrietsham
Lenham
Charing
Yalding
Sutton Valence
Headcorn
Pluckley
Bethersden
Biddenden
Cranbrook
TENTERDEN
Woodchurch
Rolvenden
Appledore
Brookland
Bodiam
Ninfield
Northiam
Peasmarsh
RYE
Brede
BATTLE
Winchelsea
Fairlight
HASTINGS
BEXHILL
Pevensey
Polegate
HAILSHAM
Herstmonceux
Hellingly
East Hoathly
Halland
HEATHFIELD
Ringmer
LEWES
Newick
Uckfield
Robertsbridge
Burwash
Mayfield
Rotherfield
Tidehurst
Hawkhurst
Frant
Harlfield
Forest Row
CROWBORDUGH
Wadhurst
Lamberhurst
Paddock Wood
Marden
SOUTHBOROUGH
TONBRIDGE
Penshurst
Edenbridge
Leigh
Hartley
Outwood
Limpfield
Charlwood
Capel
Ewhurst
Cranleigh
Ockley
Rudgwick
Horley
REDHILL
REIGATE
DORKING
Leatherhead
Ripley
Horsley
EPSOM
Banstead
CATERHAM
WARLINGHAM
Tadworth
PURLEY
Biggin Hill
ORPINGTON
Eynsford
Farningham
Darenth
Meopham
Ash
Snodland
Vorwood
Wonersh
Shalford
Bramley
Guildford
Godalming
Haslemere
Petworth
Midhurst
Pulborough
Storrington
Washington
Steyning
Findon
ARUNDEL
Angmering
WORTHING
SHOREHAM BY-SEA
BRIGHTON
HOVE
Peacehaven
NEWHAVEN
Fittleworth
West Grinstead
Cowfold
Henfield
Ditchling
BURGESS HILL
Horsted Keynes
West Hoathly
Balcombe
CUCKFIELD
HAYWARDS HEATH
BILLINGSHURST
Visborough Green
HORSHAM
CRAWLEY
EAST GRINSTEAD
Lingfield
Edenbridge
Leigh
Penshurst
Tonbridge
Paddock Wood
Marden
Sutton Valence
Headcorn
Pluckley
Bethersden
Biddenden
Cranbrook
Tenterden
Woodchurch
Rolvenden
Appledore
Brookland
Bodiam
Northiam
Peasmarsh
Rye
Brede
Winchelsea
Fairlight
Hastings
Bexhill
Pevensey
Polegate
Hailsham
Herstmonceux
Hellingly
East Hoathly
Halland
Heathfield
Ringmer
Lewes
Newick
Uckfield
Robertsbridge
Burwash
Mayfield
Rotherfield
Tidehurst
Hawkhurst
Frant
Harlfield
Forest Row
Crowborough
Wadhurst
Lamberhurst
Paddock Wood
Marden
Southborough
Tonbridge
Penshurst
Edenbridge
Leigh
Hartley
Outwood
Limpfield
Charlwood
Capel
Ewhurst
Cranleigh
Ockley
Rudgwick
Horley
Redhill
Reigate
Dorking
Leatherhead
Ripley
Horsley
Epsom
Banstead
Caterham
Warlingham
Tadworth
Purley
Biggin Hill
Orpington
Eynsford
Farningham
Darenth
Meopham
Ash
Snodland
Vorwood
Wonersh
Shalford
Bramley
Guildford
Godalming
Haslemere
Petworth
Midhurst
Pulborough
Storrington
Washington
Steyning
Findon
Arundel
Angmering
Worthing
Shoreham by-Sea
Brighton
Hove
Peacehaven
Newhaven
Fittleworth
West Grinstead
Cowfold
Henfield
Ditchling
Burgess Hill
Horsted Keynes
West Hoathly
Balcombe
Cuckfield
Haywards Heath
Billingshurst
Visborough Green
Horsham
Crawley
East Grinstead



Raster Size	350
Bit plane 1 priority: B roads	0.3
Bit plane 2 priority: A roads	1.0
Bit plane 3 priority: Main routes	1.5
Bit plane 4 priority: Motorways	2.5
Bit Plane 5 priority: Settlements	999.0
Over edge priority	1.0
Line label filter	T
Line label buffer zone	1.0
1st preferred point position	1
2nd preferred point position	13
3rd preferred point position	9
4th preferred point position	14
5th preferred point position	16
6th preferred point position	10
7th preferred point position	12
8th preferred point position	2
9th preferred point position	8
10th preferred point position	15
11th preferred point position	11
12th preferred point position	3
13th preferred point position	7
14th preferred point position	5
15th preferred point position	4
16th preferred point position	6
1st preferred line position	1

2nd preferred line position	16
3rd preferred line position	2
4th preferred line position	15
5th preferred line position	3
6th preferred line position	14
7th preferred line position	4
8th preferred line position	13
9th preferred line position	5
10th preferred line position	12
11th preferred line position	6
12th preferred line position	11
13th preferred line position	7
14th preferred line position	10
15th preferred line position	8
16th preferred line position	9
Label buffer (character widths)	T
Label vertical buffer	0.6
Label horizontal buffer	0.8
Radius of proximity (char. widths)	T
Radius of proximity	1.1
Densespace threshold % (x10)	500
Label splitting threshold %	20.0
Label not split	F
Label justification	3
Attempts before weighting label	2
Attempts before fixing label	5

Fig 6.60 Ordnance Survey 1:625000 Route Planner map database library grid square 501 - Line label preferred positions at ends of line.

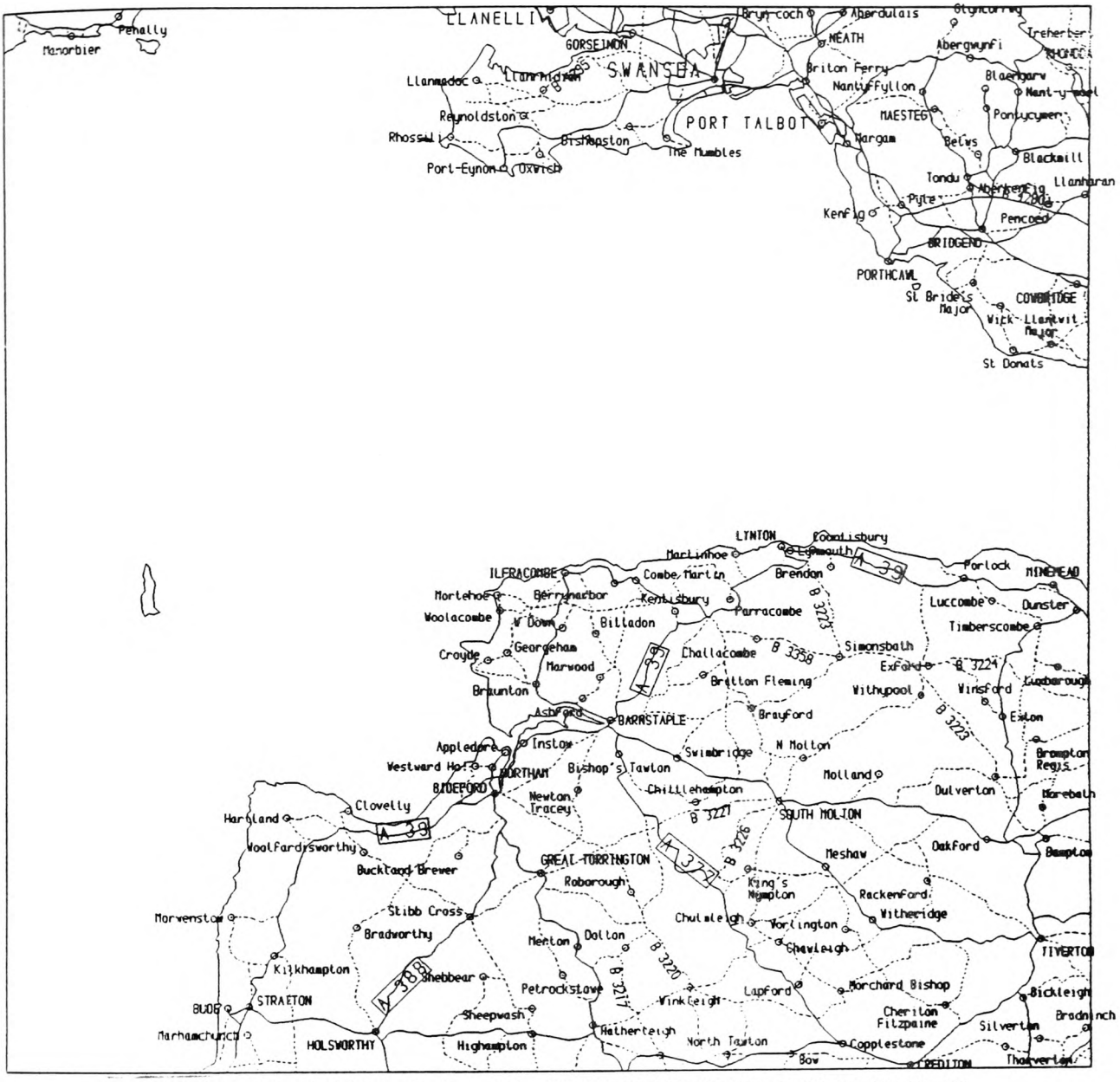
Fig 6.61 shows how the map appears when road labels are increased in size by 50%. Although large labels improve clarity, they also occupy more map space and so more conflicts result. In the case of road labels this is counteracted to some extent by the selection of a smaller number of labels.

Finally in order to enable different types of map to be produced, the user can select which features will be labelled (Rule [6.4]). The only labelled features in Fig 6.62 are settlements and airports.

6.9.4 SECTION SUMMARY

i The raster size and dense space threshold have an appreciable effect on the quality of placement. Increasing the dense space threshold (Section 6.5.3) reduces the number of label conflicts and overlaps, but at the expense of covering up underlying detail.

ii The weighting of labels should be introduced early in the iterative procedure and the maximum number of placement attempts before fixing labels in position should be sufficient (roughly twice the number of placement attempts before weighting) to let weighting take effect.



Feat. Code	Token	Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n		1600	3	0	0	0	0	0	0
1	mway_service_ltd_e		1600	3	0	0	0	0	0	0
2	mway_service_ltd_s		1600	3	0	0	0	0	0	0
3	mway_service_ltd_w		1600	3	0	0	0	0	0	0
20	motorway		600	3	1215	938	810	303	1970	1350
21	motorway_junction		1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs		1600	3	0	0	0	0	0	0
23	mway_service_area		1600	3	0	0	0	0	0	0
30	primary_route_s_c		450	1	1215	938	810	303	1875	1200
31	primary_route_d_c		550	2	1215	938	810	303	1875	1200
32	prim_route_narrow		400	1	1215	938	810	303	1875	1200
35	prim_route_srv_area		1600	1	0	0	0	0	0	0
40	main_road_s_c		300	1	1215	938	810	303	1875	1200
41	main_road_d_c		550	2	1215	938	810	303	1875	1200
42	main_road_narrow		300	1	1215	938	810	303	1875	1200
49	b_road		125	0	900	657	600	225	938	657
50	other_road		125	0	0	0	0	0	0	0
61	city		5000	4	1075	750	720	269	1375	1040
62	large_town		3000	4	1075	688	720	269	1312	938
63	village_on_prim_rt		1500	4	850	600	500	188	1125	625
64	town_on_prim_route		1500	4	1075	688	720	269	1312	888
65	small_town_village		1500	4	850	600	500	188	1125	625
75	airport		1450	1	900	700	500	188	1100	900

Fig 6.61 Ordnance Survey 1:625000 Route Planner map database library grid square 201 - Road labels 50% bigger.

Feat. Code	Token	Description	Feat. Width	Bit Plane	LT_HT	LT_WT	LW_CS	LT_DS	BL_HT	BL_WT
0	mway_service_ltd_n		1600	3	0	0	0	0	0	0
1	mway_service_ltd_e		1600	3	0	0	0	0	0	0
2	mway_service_ltd_s		1600	3	0	0	0	0	0	0
3	mway_service_ltd_w		1600	3	0	0	0	0	0	0
20	motorway		600	3	0	0	0	0	0	0
21	motorway_junction		1600	3	0	0	0	0	0	0
22	mway_jun_ltd_acs		1600	3	0	0	0	0	0	0
23	mway_service_area		1600	3	0	0	0	0	0	0
30	primary_route_s_c		450	1	0	0	0	0	0	0
31	primary_route_d_c		550	2	0	0	0	0	0	0
32	prim_route_narrow		400	1	0	0	0	0	0	0
35	prim_route_srv_area		1600	1	0	0	0	0	0	0
40	main_road_s_c		300	1	0	0	0	0	0	0
41	main_road_d_c		550	2	0	0	0	0	0	0
42	main_road_narrow		300	1	0	0	0	0	0	0
49	b_road		125	0	0	0	0	0	0	0
50	other_road		125	0	0	0	0	0	0	0
61	city		5000	4	1075	750	720	269	1375	1040
62	large_town		3000	4	1075	688	720	269	1312	938
63	village_on_prim_rt		1500	4	850	600	500	188	1125	625
64	town_on_prim_route		1500	4	1075	688	720	269	1312	888
65	small_town_village		1500	4	850	600	500	188	1125	625
75	airport		1450	1	900	700	500	188	1100	900

Fig 6.62 Ordnance Survey 1:625000 Route Planner map database library grid square 301 - No road labels.

iii The label splitting threshold has little effect on the number of label conflicts and overlaps and so is more applicable to map design where the splitting of labels may be encouraged for aesthetic reasons.

iv Using a large radius of proximity for point labels (approximately one character width) results in a small reduction of label conflicts but increases ambiguity slightly.

v The X buffer distance has an uncertain effect on label conflicts, however the Y buffer increases conflicts if made too large and so should be set to a small value. However both the X and Y buffer should be given adequate values to avoid label ambiguity (X buffer=0.8 and Y buffer=0.6).

vi Ambiguity can also be reduced by the use of suitably sized settlement symbol widths for the raster image, so that such symbols are slightly larger than the label radius of proximity.

vii LABPOS provides the map user with good control over the map design in the aspects of label splitting, map feature importance, label selection on a feature code basis, line label selection according to line length, preference of placement positions and label sizes. This has been demonstrated by the illustrations in this section (Figs 6.52 to 6.62).

6.10 CONCLUSION

Through the use of LABPOS, it has been possible to investigate new and existing name placement techniques. These have been applied to the Ordnance Survey Route Planner map which is quite a complicated map to label compared to the maps tackled by previous automated name placement systems (Chapter 4). The new techniques that were investigated include an iterative name placement algorithm, raster bit planes, optimization and fifteen user controllable rules and specifications. The existing techniques investigated included constrained (parameterized) label positioning algorithms, the use of weights or priorities to indicate the preferences of different label positions, the use of raster data to detect overlaps, the use of lists of possible positions and potential overlaps for labels, and the use of different label configurations.

The success rate for LABPOS, which varied according to cartographic feature density, could be measured in two ways: by the number of labels not in conflict and the number of labels not in overlap. Approximately 80% of labels could be placed successfully without being in conflict and about 90% without being in overlap with each other. LABPOS was able to achieve a name placement rate of approximately one label every one and a half seconds (This includes database access and rasterization times) and so

is over a hundred times faster than manual placement (Section 1.1).

Both the speed and the placement success rate of LABPOS should make the program of direct use to the Ordnance Survey for the generation of the Route Planner map and any new map series produced using the same database structure. A copy of LABPOS has been installed on the Ordnance Survey's VAX11/750 computer in their research and development department (Southampton) with the intention of being applied to future maps (King, 1987). When LABPOS is integrated into map production, some manual interaction will be required to reposition those few remaining labels which had not been placed satisfactorily. This would slow down the overall placement rate somewhat, but by using interactive editing it should still be faster than manual placement.

There are several limitations with LABPOS, for instance it was not designed to position area names or curved river names. Such features are however unlikely to be amended in different editions of the Route Planner map, unlike newly built roads. Therefore it may be possible to consider area and curved line labels as underlying map features of very low priority and include these in the cartographic data so that LABPOS could be tricked into thinking that these are features rather than labels.

Other aspects of LABPOS which require attention are rasterization and record searching which could be performed more efficiently. The graph in Fig 6.15 shows that the LABPOS CPU time for different raster sizes, ranging from very small (25x25) to large (700x700), takes up between a third and a quarter of the total program CPU time. Therefore rasterization would appear to make only a relatively small contribution to the overall CPU time. The process in LABPOS which is the most time consuming (as observed from running the program) is the label conflict resolution strategy. This involves repeatedly reading from or writing to the "LABEL1" file. If the "LABEL1" file were to be stored in an unformatted binary form then this would allow a faster access time to the records and a shorter overall program CPU time.

Although not strictly to do with LABPOS, the name placements on maps generated using LABPOS have so far only been displayed in black and white. Unsharp masking is used on many paper maps to make labels appear distinguishable from underlying detail (O.S., Anon). If unsharp masking were used in the display of the LABPOS label placements then it would be possible to increase the "dense space threshold" without affecting the ability to discern labels from underlying detail and hence reduce the number of label conflicts.

Another limitation of LABPOS is that it was specially designed with one particular map and database in mind. This is a restriction of many name placement systems developed in the past. A logic programming approach as discussed in chapter one provides a solution to this problem. This would allow the implementation of name placement rules, such as those mentioned in section 2.2, which are more complicated than those used in LABPOS. Logic programming also has an important advantage in that the development time is considerably less than that for LABPOS which took over a year to write, debug and optimize. The next chapter (Chapter 7) goes on to discuss such an experimental system as developed by the author. It incorporates a special name orientated, topological and spatial cartographic database which is capable of supplying information needed by high level name placement rules. Many of the techniques investigated in LABPOS have been incorporated into the design of the new name placement system, these include:

1) Raster bit planes, feature masking and feature class priorities.

2) Parameterized point and line label positioning algorithms.

3) Different sized labels and different label configurations i.e. label splitting.

4) Generation of potential label:label overlap data and a list of available positions for each label.

5) Label:label overlap detection using a technique based upon the Cohen-Sutherland clipping algorithm and an allowance for a variable buffer zone.

CHAPTER 7

"NAMEX" - A RULE-BASED NAME PLACEMENT SYSTEM

7.1 INTRODUCTION

7.1.1 SYSTEM DEFINITION

In the present and subsequent chapter, an account is given of the author's application of logic programming techniques to the development of rule-based name placement systems. Unlike LABPOS (Chapter 6), rule-based systems are encoded at a much higher level, can be more concise, generally require less development time, and are capable of handling a wider range of maps. The language of PROLOG (Section 1.3) will be used for implementing the rule-based systems because it is particularly suitable for encoding high level rules and strategies. However database access and low level name placement facilities will be catered for by a FORTRAN program which is more efficient at these tasks than the equivalent in PROLOG.

The present chapter describes NAMEX, a structured programming environment which consists of a cartographic database adapted to name placement and primitives (PROLOG predicates) which are used to query the database and handle name placement (via FORTRAN). Chapter 8 illustrates

how these primitives can be used to perform name placement on a wide range of maps. However because of practical considerations, such as the development time for the whole of the NAMEX system only a small selection of the label configurations and positions discussed in chapter 5 have been implemented.

The NAMEX system (Fig 7.1) consists of seven components:

i. The source cartographic database.

ii. A derived name placement database.

iii. A cartographic data conversion program which generates (ii) using (i).

iv. A facility to query the name placement database and handle name placement through the use of primitives.

v. A name placement logic program consisting of a strategy and rule-base.

vi. An interface to the name placement system using (iv), (v) and a user menu.

vii. A program to display the annotated map.

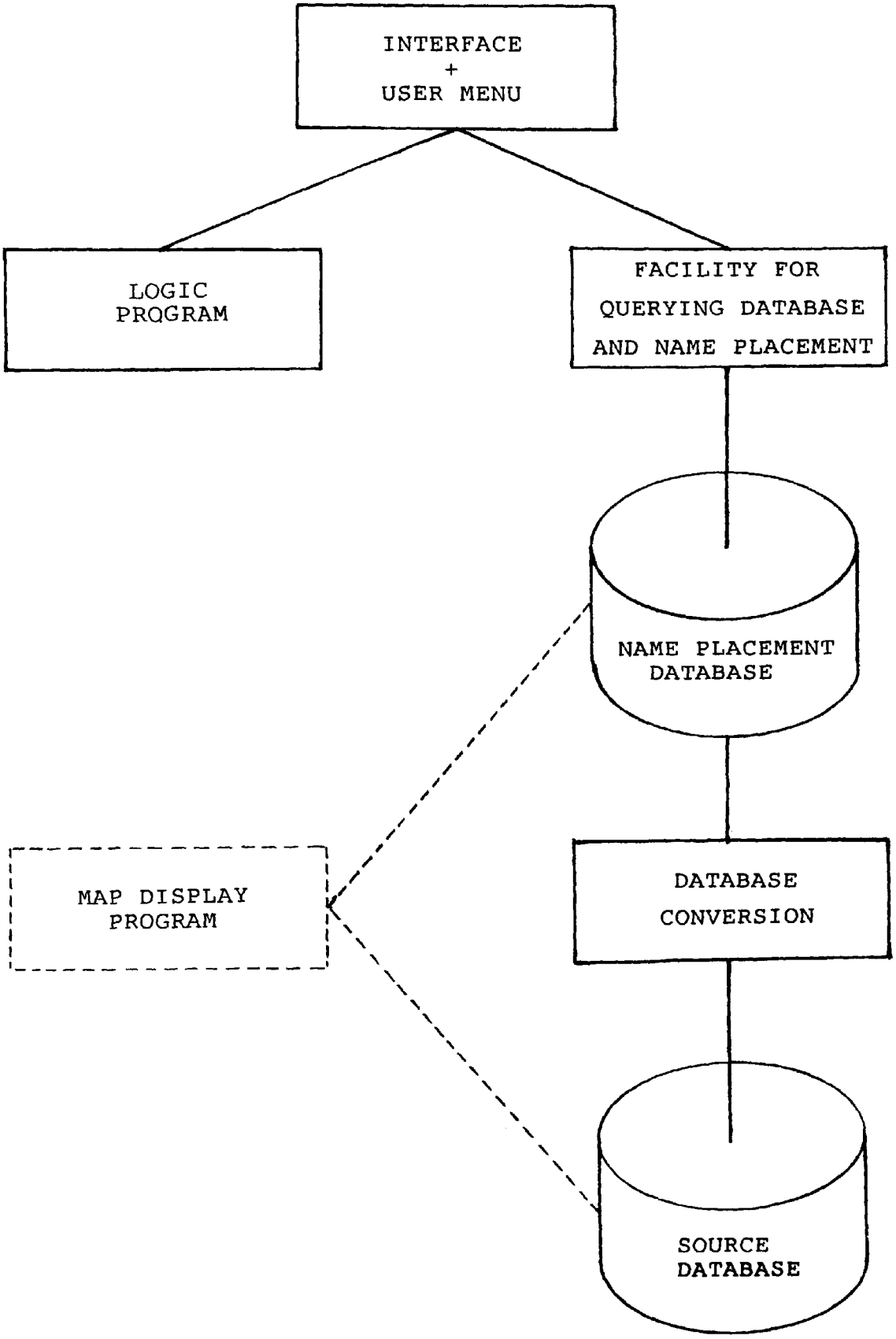


Fig 7.1 The NAMEX name placement system.

To construct a name placement system using NAMEX, the user must supply a data conversion program (iii) and a map display program (vii), execute the data conversion program to generate the derived name placement database (ii), and define the logic program strategy and rule-base (v). Once these have been constructed for a particular map type, the user need only be concerned with the interface (vi) for actual name placement.

Three levels of user are anticipated for the NAMEX system. Firstly the basic user who will want to carry out name placement with an existing name placement system using simple menu options via the interface. Secondly the knowledge engineer, whose job it will be to update the logic program, with new rules and/or a new name placement strategy (using existing primitives) designed for a specific map requirement. Finally, the system programmer who can add new primitives for database querying and name placement handling.

7.1.2 THE DATABASE

A name placement database has to provide all the necessary information required to perform name placement. Because labels are the objects to be placed, it should be designed to be a name rather than feature orientated system, but retain enough topological information to

answer spatial queries (Figs 7.2 and 7.3). The Ordnance Survey Route Planner map database used by LABPOS (Chapter 6), was relatively slow at extracting high level information such as finding out which links make up a road of a given name. This was because it was essentially a feature classified database.

A prime consideration in the design of the database should be speed of access to information and data compaction. Compaction techniques should be utilised, especially in parts of the database which are likely to consume the most storage space, but not at the expense of efficiency or ease of detectability of errors during name placement system development.

Different maps are often stored in different data structures, therefore in order to be able to place names on any kind of map, it is necessary to format a copy of the source cartographic data into the NAMEX database using a program which must be written for this purpose by the user. The NAMEX data structure was designed to be fairly simple so that cartographic data conversion can be performed relatively easily.

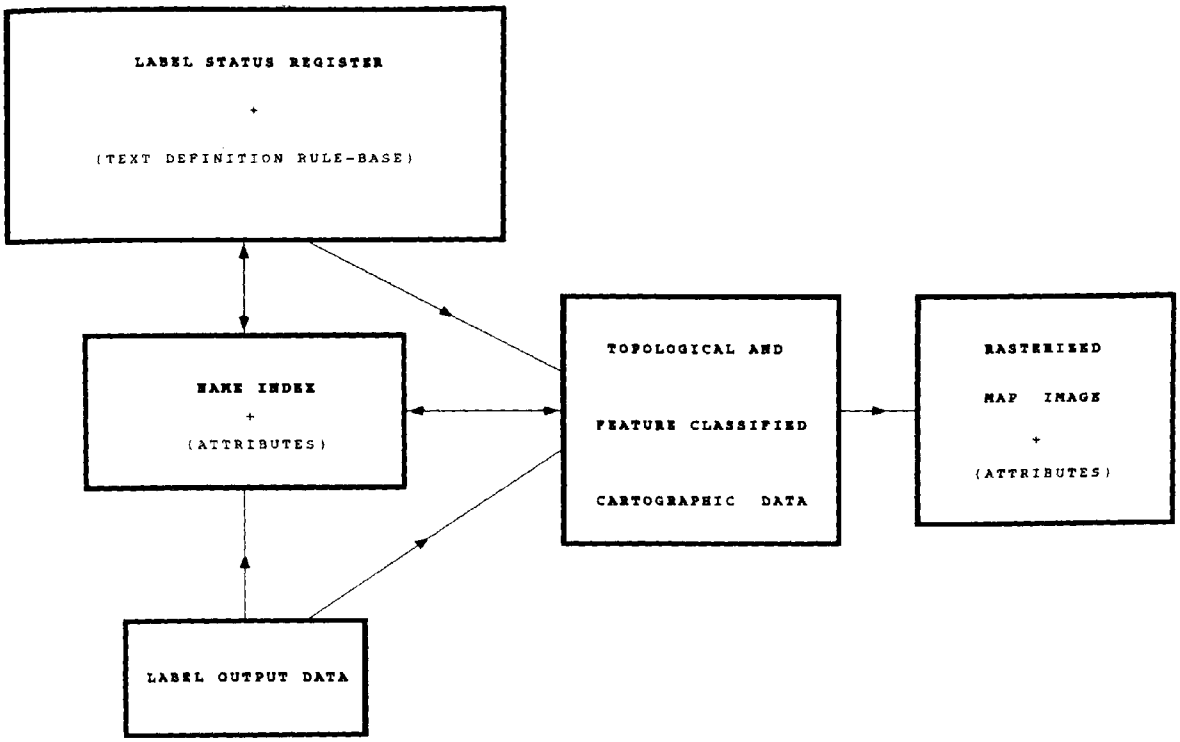


Fig 7.2 Schema of the name placement database.

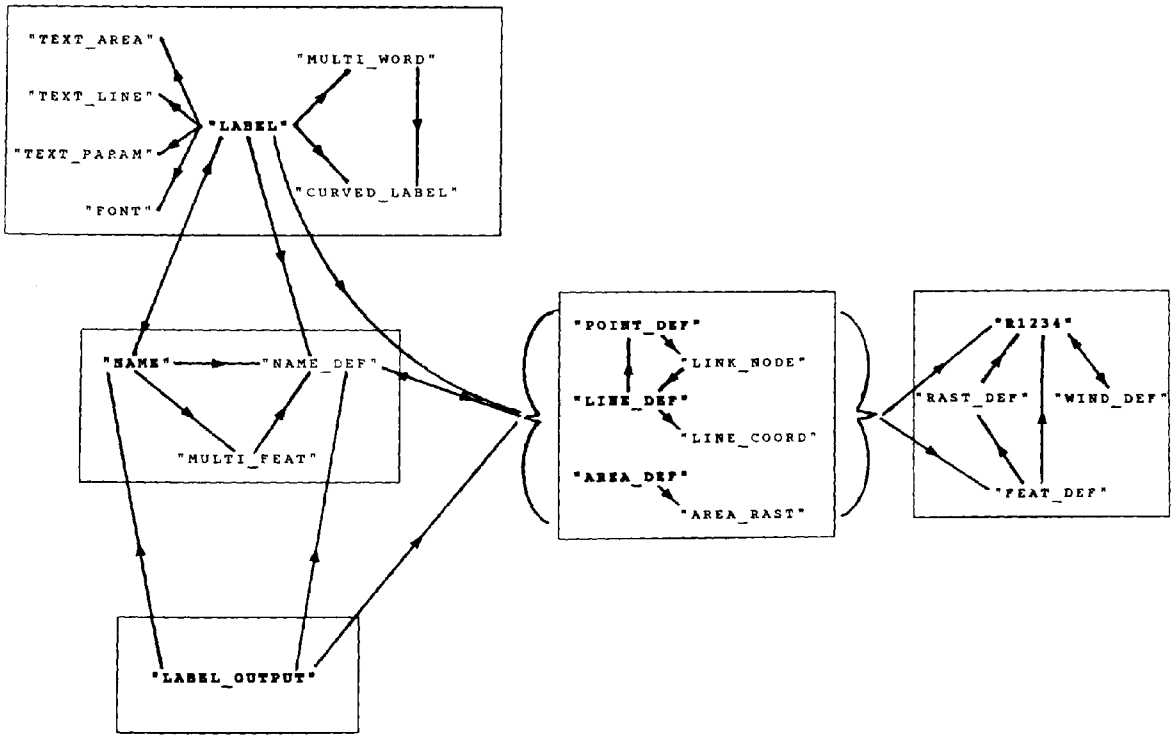


Fig 7.3 Schema of the relationship between files in the name placement database.

The database consists of the following components:

i. Name index and attributes.

ii. Topological and feature classified cartographic data.

iii. Raster map image.

iv. Label status register and text definition rule-base.

v. Label output data.

Access to information in the database pertaining to name placement is made through the name index files. These contain information such as a name and pointers to multiple occurrences of that name, should they exist. Other attributes include the code and type (point, line or area) of feature that a particular name represents and a pointer to it in the topological and feature classified cartographic data.

The topological and feature classified cartographic data are stored in files related to feature type. Most of the data are in vector form apart from the area data which are held in a run-length encoded raster format. The

topology between point and line features is embedded in the data structure and pointers are available to relate back to the name index if necessary. Many cartographic features present in the data do not have names, but are retained for topological and raster purposes.

The raster map image in the database consists of several small run-length encoded files which make up grid squares of the whole map image. When these files are loaded, the raster map image is reconstituted in the form of an array. The size of the grid squares, the scale and window parameters for the raster image, and the raster attributes for the map image are stored in other files.

A label status register is used to keep a record of all current label positions, configurations, sizes etc. It is constructed using the names contained within the name index according to selection criteria defined in the strategy. It consists of three files which are designed to record the status of single lined, split or spaced out (curved) labels. In practice though, only single lined labels are handled in the current NAMEX system. To help decide which positions, configurations or sizes are available for labels associated with particular feature codes, a text definition rule-base is available (contained in several files). This is defined through interactive editing via the user interface.

Once all the labels have been placed, the contents of the files recording the status of each label, are used to generate a label output data file.

7.1.3 THE PROGRAMS

The NAMEX system consists of a mixture of PROLOG programs which perform high level tasks and FORTRAN programs which perform low level tasks. Documentation for these programs is available in a separate volume to this thesis.

The PROLOG programs are:

i. NAMEX, the name placement system interface with a high level user menu.

ii. LOGIC, the strategy and rule-base. The contents of this vary according to map type.

The FORTRAN programs are:

i. DB_GENERATE, a cartographic data conversion program.

ii. DB_ACCESS, a facility to query the derived name placement database and handle name placement through the use of primitives.

iii. DB_DEF, a low level user menu for defining raster window size, raster and feature attributes, and permissible label positions / configurations / sizes in the parameterized rule-base.

iv. DB_PLOT, a cartographic display program.

The NAMEX program forms an interface to all the other programs (except DB_GENERATE and DB_PLOT) and is operated via a user menu (Fig 7.4). The first menu option loads cartographic data into the system. The second option is used after the first and consults the LOGIC name placement program, selecting which labels are suitable to place (See chapter 8). Although LOGIC coordinates name placement, most of the high level primitives that it uses are defined inside NAMEX. DB_ACCESS is interfaced to NAMEX and can be commanded to query the database and perform name placement through the use of low level primitives. NAMEX can also

```
**********************************************************
**********************************************************
**     ================                                 **
**  # N - A - M - E - X #   (A name placement system)    **
**     ================                                 **
**        By Tony Cook    -  1987 Polytechnic of Wales    **
**                                                        **
**********************************************************
**********************************************************


Do you want to:

<1> Load up existing cartographic data
<2> Select labels to be placed
<3> Carry out name placement
<4> Save name placement data
<5> Edit name configuration rules/parameters
<6> Have some help
<7> EXIT
```

Fig 7.4 Introductory menu to the NAMEX name placement system.

Table 7.1 "KEEP_TRACK" record contents

Record No.	Name	Description
1	ONC	"NAME" file record count.
2	NC	"NAME_DEF" file record count.
3	PC	"POINT_DEF" file record count (Points).
4	LC	"LINE_DEF" file record count (Lines).
5	AC	"AREA_DEF" file record count (Areas).
6	NOC	"POINT_DEF" file record count (Nodes).

(This file is accessed by both FORTRAN and PROLOG).

call DB_DEF which allows editing of the text definition rule-base, feature and raster definition files (Option 5).

Option three of NAMEX activates the name placement process and the results of the placement can be saved to an output file by selecting option four. Option six is included to allow a help facility to be added at a later date, but in the present system does nothing. Finally, the user can exit to PROLOG by selecting option seven.

DB_GENERATE is operated separately from the other programs because the process of database conversion can take a long time to complete. DB_PLOT is also operated separately since the user may wish to integrate it with an existing graphics display program.

7.1.4 PROLOG

Although the database and name placement primitives mentioned in this chapter are not described in full, it will be necessary for the reader to have some basic understanding of PROLOG, if the use of these primitives is going to be understood prior to the next chapter. A brief explanation of the simple principles behind PROLOG, with examples, is given in Appendix 2. Suffice to say that to the left of the ":-" or logical implication is the predicate name or head of the logical clause, and to the right, and terminated by a ".", are goals consisting of

other predicates. The "," represents a logical "AND" implying that each subsequent goal must be satisfied if the end of the predicate is to be reached. An ";" refers to a logical OR between adjacent goals. The "!", known as the "cut", is used in order to control backtracking. Predicates sometimes have curved brackets on their righthand side, these can contain variables, atoms or constants, and lists. The difference in appearance between a variable and an atom, is that variables always start with capital letters whereas atoms start with lower case letters or are numbers. A list can easily be identified because it consists usually of a list of atoms or variables in between square brackets. The left most element of the list is termed the head, the remaining elements, the tail. A variable can either have a value (instantiated) or be without a value (un-instantiated), but will eventually be assigned one by the predicate. Finally, comment statements are sometimes placed into predicate listings, these lie between comment markers: "/*" and "*/".

7.1.5 CHAPTER CONTENTS

The chapter so far has given an overview of the NAMEX system and a brief guide to understanding PROLOG. The next section of the chapter will describe how the interface between PROLOG and FORTRAN is achieved. Because the

success of the NAMEX system depends upon the data structure, most of this chapter will describe how the database is organised and can be accessed from PROLOG. The small number of primitives needed by NAMEX for name placement will be discussed towards the end of this chapter (Section 7.8). PROLOG listings given in this chapter and the following chapter will be in either of two forms, the original listings (bold type) and pseudo-logic listings (ordinary type). The pseudo-logic listings are used where the original PROLOG listings are lengthy or are difficult to understand.

7.2 PROLOG-FORTRAN INTERFACE

Despite the use of the high level language PROLOG for controlling the system, both the database and the low level components of the primitives are stored externally for memory and efficiency reasons and are controlled from a low level FORTRAN program, DB_ACCESS. A means of accessing the data and the low level primitives from PROLOG must be established.

The PROLOG primitive predicate, "set_up" (Appendix 3), externally loads and links NAMEX with the FORTRAN object code of programs DB_ACCESS and DB_DEF. It provides the interface between PROLOG and FORTRAN through the use of the PROLOG predicate "command", with its associated FORTRAN counterpart subroutine "COMMAND", in DB_ACCESS. Most high level primitives in the NAMEX system will make use of "command" at their lowest level.

The access of the COMMAND subroutine from the higher level PROLOG language, is possible thus:

```
command(P1,P2,P3,P4,P5,P6,Out).
```

This makes use of up to six input integer parameters and one output integer parameter. The parameter P1 is the index number of the primitive being called, the remaining parameters are either sub-primitive index numbers, record

numbers or attributes which coordinate name placement (Appendix 3). The example primitive below makes use of "command" to find the length of a line feature:

```
get_line_length(N,Length) :-
    valid_feat(N),
    command(32,N,2,0,0,0,Length), !.
```

In this example, the primitive "valid_feat" (Appendix 3) checks that the feature is valid. If so, then "command" indexes primitive number 32 in the "COMMAND" subroutine in DB_ACCESS. The "N" refers to the record number of the line and the "2" to the field number whose contents, the line length, are returned as "Length" (Section 7.4.3).

In order to be able to pass an array of data from FORTRAN to a list in PROLOG, the data is initially stored in a section of a thousand element integer memory array in the FORTRAN DB_ACCESS program. Then, the individual sequential elements of the array are examined in turn and appended together to form a list of the specified length in PROLOG, using the PROLOG primitive:

```
get_list(List_length,List).
```

Similarly, an array of data can be passed from PROLOG to FORTRAN by using the PROLOG primitive:

put_list(List).

Likewise, additional use is made of uniquely named variables in FORTRAN, which are associated with the fields of each database file. When calling a primitive to read or write the contents of a file, if a record number is given in the primitive then the complete record is read or written to and all the unique field variables are updated with new contents. If the record number is not given, then the primitive will only access the current field variable contents.

Another way that PROLOG can be interfaced to FORTRAN, albeit indirectly, is through the use of files which are readable in both languages. Table 7.1 shows the record contents of the "KEEP_TRACK" file which is generated in FORTRAN and indicates the number of records present in files in the NAMEX database. Once these records have been read into the PROLOG NAMEX program, they are asserted as PROLOG facts for later use:

```
name_count(Onc).

name_def_count(Nc).

point_count(Pc).

line_count(Lc).

area_count(Ac).

node_count(Noc).
```

7.3 THE NAME INDEX

7.3.1 INTRODUCTION

The most important aspect of the name placement system is its name information and for that reason an index is kept of all known names on the map, whether or not these are eventually applied as labels. The design of the index is complicated by the fact that, although one name will usually uniquely identify a single map feature, there are exceptions. For instance, a single name can be associated with many features, such as the "ISLES OF SCILLY", which applies to more than a dozen closely associated islands and rocks (Fig 2.8). On other occasions, two or more completely separate features have identical names, but are not physically associated (Note that there are two "Llantrisant"'s in the upper left half of Fig 2.19). Finally, there are some features with more than one label, such as mountain peaks which usually have both a spot height label and the name of the mountain. The structure of the name index must allow for such relationships.

7.3.2 NAME INDEX FILES

The database name index consists of the "NAME" file which keeps a record of each name, "NAME_DEF" which

provides additional attributes about each name and its associated cartographic data, and "MULTI_FEAT" which allows the one to many relationship between names and features to be represented (Table 7.2, 7.3, 7.4 and Fig 7.5). Whether identically named features are related can be deduced if they share common nodes as in the case of lines, or if their spatial proximity to each other is less than say a threshold distance. In a logic program, the threshold distance could form part of the information required to decide which named features are suitable for labelling.

In case backtracking occurs during the name placement process and the record of the name held in the label status register undergoes modification, such as abbreviation, an original version of each unique name is kept in the "NAME" file. In the "NAME" file, the field NAME_NO1 indicates the number of features that the name corresponds to. If the name is unique, then a pointer, NAME_POINT1, is used to point directly to the "NAME_DEF" file for further information. If a name is not unique, then the pointer refers to other occurrences of features of the same name, whose location in the "NAME_DEF" file (not necessarily adjacent) can be found by looking at the additional pointer file "MULTI_FEAT".

In the "NAME" file, the second pointer field, NAME_POINT2, and the number of occurrences field,

Table 7.2 "NAME" record structure

Field name (No)	Description
NAME (1)	Label name.
LET_COUNT (2)	No. of characters in name (including spaces).
NAME_POINT1(3)	"NAME_DEF" / "MULTI_FEAT" pointer.
NAME_NO1 (4)	No. of occurrences of name.
NAME_POINT2(5)	Selected label pointer.
NAME_NO2 (6)	No. of occurrences of selected label.

Table 7.3 "MULTI_FEAT" record structure

Field name	Description
MULTI_NAME	Pointer to occurrence of name in "NAME_DEF".

Table 7.4 "NAME_DEF" record structure

Field name (No)	Description
FCODE (1)	Feature Code.
FTYPE (2)	Feature Type (0,1 or 2).
FPOINT (3)	Feature Pointer.
NPOINT (4)	Name Pointer.

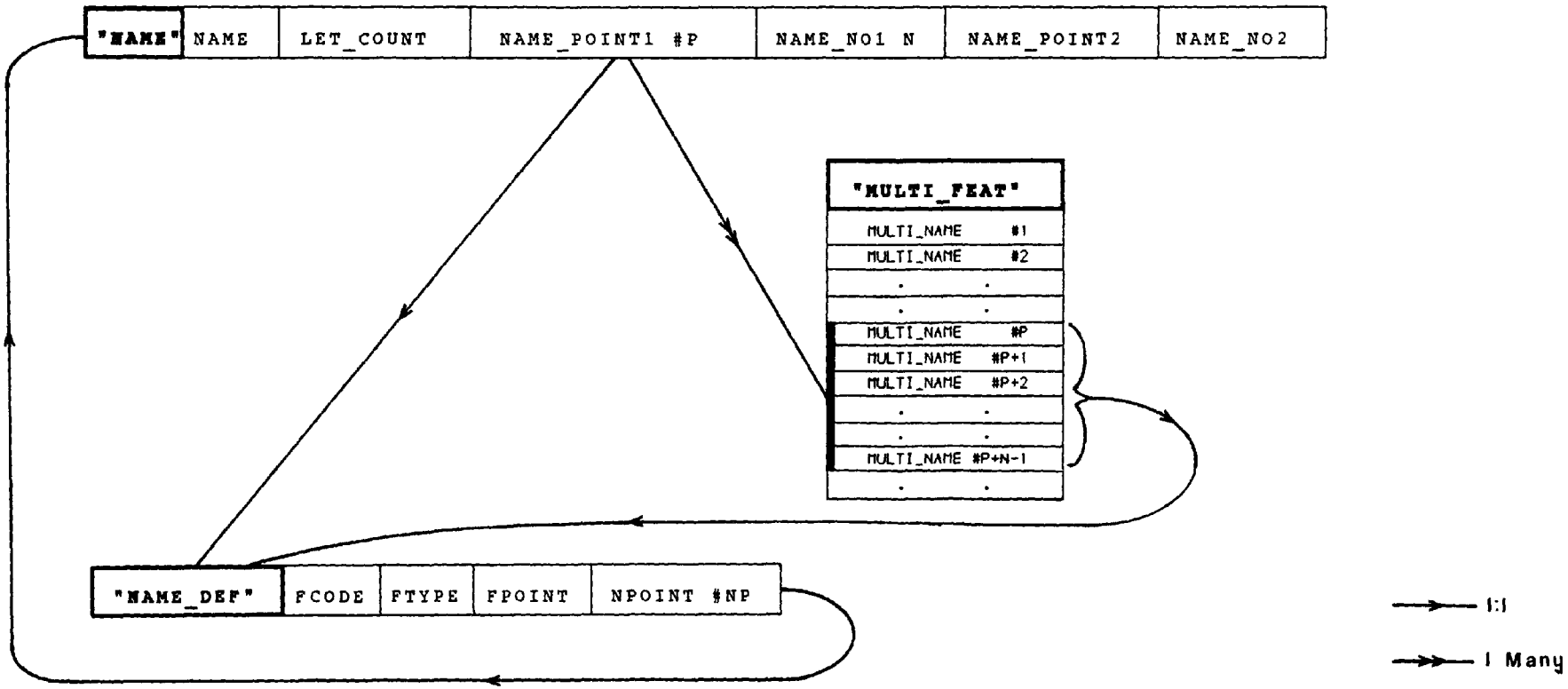


Fig 7.5 The name index - mutiple name extraction.

NAME_NO2, are intended to point to the location in the label status register which records labels that have been selected for placement. However, providing that same named labels are contiguously selected in the label status register, there is no need for an additional pointer file. In fact in the current implementation of the NAMEX system, NAME_POINT2 and NAME_NO2 are not used.

The "NAME_DEF" file contains name attribute data which comprise of a feature code, a feature type (Point, line or area) and a feature pointer, which points to cartographic feature information held in topological and feature classified data files. The feature code is a number with a value between 0 and 999 and which uniquely describes the class of feature that is being represented. All feature codes are defined in the "FEAT_DEF" file (Table 7.12). The feature code and type fields are included in the "NAME_DEF" file, rather than in "NAME", because the name may refer to several features whose classes are dissimilar. The feature pointer, FPOINT, points to the associated feature record in a cartographic data file, determined by feature type. A name pointer field, NPOINT, is also included, in case the need arises to refer back to the main "NAME" index file.

7.3.3 NAME INDEX PRIMITIVES

PROLOG primitives exist for accessing the fields of the name index files. The name index primitives, for files "NAME" and "NAME_DEF", are of the form:

```
get_name_....
get_name_def_....
```

For example, to access the ASCII code of the N'th letter in a name belonging to record I in the "NAME" file, one would use:

```
get_name_asc_value(I,N,Asc).
```

In fact, using the above predicate, a higher level recursive predicate, can be written which extracts whole names in the form of strings:

```
grab_name(Rec,Text):-
   get_name_letter_length(Rec,Let_count),
   grab_name2(Let_count,[],List),
   name(Text,List), !.
   /* "name" converts a list of ASCII numbers into
      a string (A POPLOG library predicate) */
```

```
grab_name2(0,List,List):- !.
   /* Recursively builds a list until all letter
      positions have been examined */

grab_name2(Let_pos,List,Result):-
   get_name_asc_value(Let_pos,Char),
   append([Char],List,New_list),
   /* "append" appends one list to another to form
   a new list (See appendix 2) */
   New_pos is Let_pos-1,
   grab_name2(New_pos,New_list,Result).
```

A similar job is performed by another predicate, but for the purpose of placing ASCII codes values of the characters in a name into FORTRAN memory for labelling purposes:

```
put_text_into_mem(Name_num).
```

Two general purpose primitives:

```
read_name_details(Rec,Field_no_list,Parameter_list).

read_name_def_details(Rec,Field_no_list,Parameter_list).
```

are available in order to retrieve whole lists of parameters from the associated index files, by specifying a list of the field numbers. For example to read the feature code (Field 1), the feature type (Field 2), and feature pointer (Field 3) of "NAME_DEF" record 42, the following is called:

read_name_def_details(42,[1,2,3],[Fcode,Ftype,Fpoint]).

The primitive:

get_names(Multi_name_ptr,No_of_names,Name_def_ptr).

is provided so that, on being forced to backtrack "No_of_names" times, it returns all the pointers to the "NAME_DEF" file. This alleviates the need for the user to be aware of the presence of the "MULTI_FEAT" file.

To check that the name index records being used are within range, validation can be performed with:

valid_name(Name_rec).

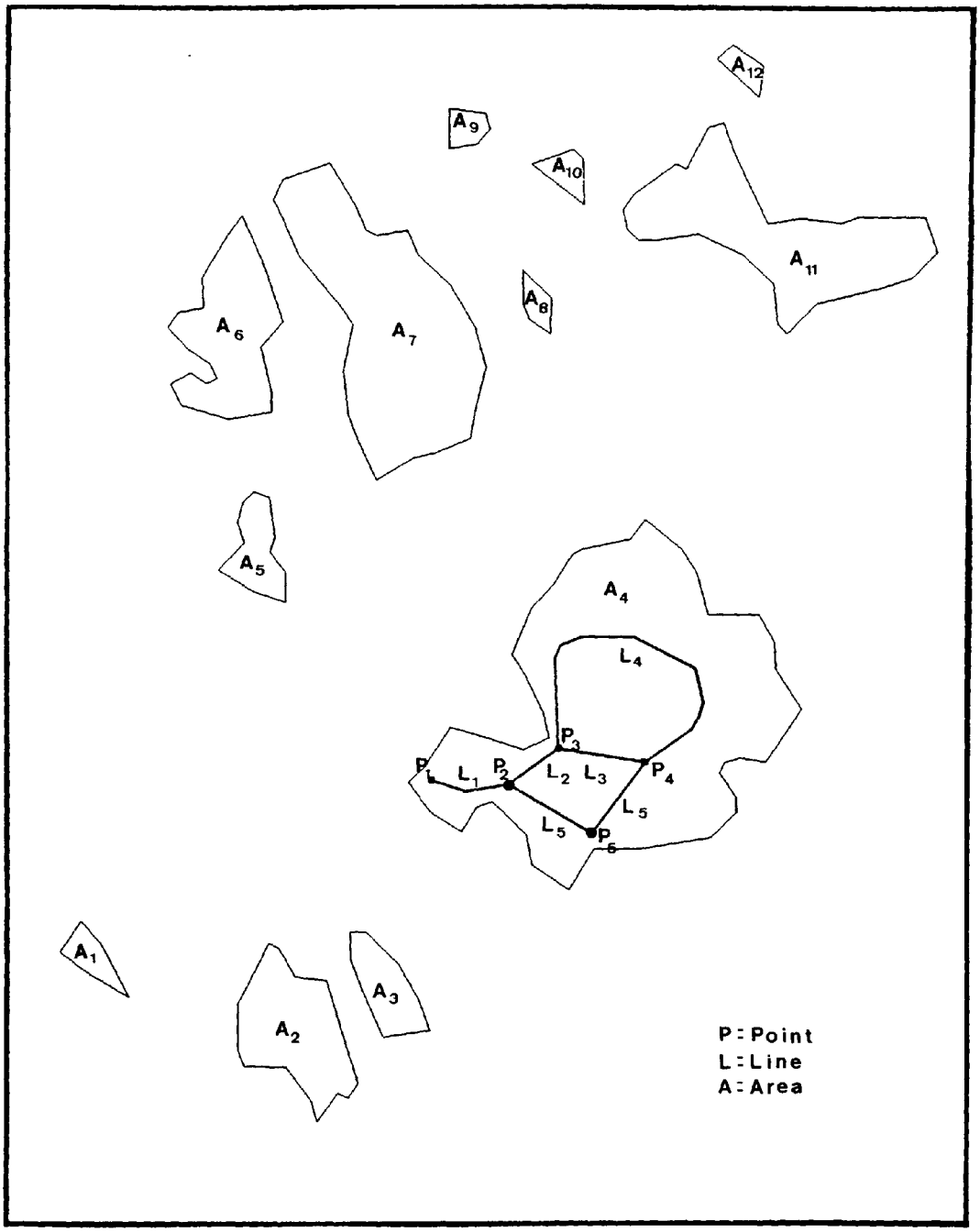
valid_name_def(Name_def_rec).

Finally, the process of selecting names from the "NAME_DEF" file suitable for labelling is performed with:

```
valid_label(Name_def_rec):-
   get_name_def_fcode(Name_def_rec,Fcode),
   /* Gets recommended font number for name
      (See section 7.6.5) */
   get_text_param_font(Fcode,Font),
   /* checks that font number is not zero i.e.
      label not to be placed (See section 7.6.5) */
   valid_font(Font),
   get_name_def_ftype(Name_def_rec,Ftype),
   /* Final label selection test (See section 7.8.6) */
   name_select(Name_def_rec,Ftype), !.
```

7.3.4 SECTION SUMMARY

This section of the chapter has described the structure of the name index and some of the primitives used to access it. A graphical portrayal of the name index structure as applied to the "Scilly Isles" can be seen in Fig 7.6. A more detailed description of the name index primitives can be found in appendix 3. The next section shows how to obtain topological and feature classified data pertaining to a name.



Record No.	"NAME" file			"NAME_DEF" file			"MULTI_FEAT" file
	NAME	NAME_DEF Pointer	No. of records	FSN	FTYPE	NAME Pointer	NAME_DEF Pointer
1	Scilly Isles	1	12	1	2	1	1
2	St. Marys	13	1	2	2	1	2
3	Tresco	14	1	3	2	1	3
4	St Agnes	15	1	4	2	1	4
5	St Marys	16	1	5	2	1	5
6	Hugh Town	17	1	6	2	1	6
7	-	-	-	7	2	1	7
8	-	-	-	8	2	1	8
9	-	-	-	9	2	1	9
10	-	-	-	10	2	1	10
11	-	-	-	11	2	1	11
12	-	-	-	12	2	1	12
13	-	-	-	13	2	2	-
14	-	-	-	14	2	3	-
15	-	-	-	15	2	4	-
16	-	-	-	1	0	5	-
17	-	-	-	2	0	6	-

Fig 7.6 Representation of the structure of the name index for the Scilly Isles.

7.4 TOPOLOGICAL AND FEATURE CLASSIFIED DATA

7.4.1 INTRODUCTION

The topological and feature classified data serve two purposes, firstly as a source of spatial knowledge and secondly as temporary masks for specific features in the raster image during the testing of name placement positions (Sections 6.5.3 and 7.5.5).

Most features have a name attribute. However those which do not must be retained for completeness of the topological structure. There are also some features which although not plotted on the map are labelled, such as bays and channels. The location and spatial extent of such features must be retrievable in order to determine where to place their labels and this is achieved with the use of approximate areal data.

All features in the topological and feature classified data have a feature structure which includes a feature code and a feature type, similar to the Ordnance Survey 1:625000 scale database. The feature code which is used in the name index, for quick reference, must also be duplicated in the topological and feature classified data because not all features have names and the codes, of those that do not, are required for rasterization purposes.

7.4.2 POINT DATA

The "POINT_DEF" file contains information concerning cartographic point features, which are either named or unnamed points, or featureless nodes (Table 7.5). Featureless nodes are given a feature code value of zero. The total number of valid point records, some of which are also nodes, is given by PC in the "KEEP_TRACK" file (Table 7.1). The remaining records PC+1 to NOC contain just node records.

The point features can be indexed from both the name index (Fig 7.7) and the line data, if the points concerned are nodes (Fig 7.9), and vice versa.

7.4.3 LINE DATA

Line features can be accessed from the name index (Fig 7.8) and by point features via the "LINK_NODE" file (Table 7.8). The "LINK_NODE" file provides a pointer to line records associated with a particular point or node. By using the end node pointers to refer back to the point file, and using the "LINK_NODE" file, it is possible to find the record numbers of all connected lines. This determines the one to many relationship between points/nodes and lines (Fig 7.9).

Table 7.5 "POINT_DEF" record structure

Field name	Description
FCODE	Feature Code
PEAST	Point Easting (m)
PNORTH	Point Northing (m)
LPOINT	"LINK_NODE" pointer
LNO	No. of links at point
N_POINT	"NAME_DEF" pointer

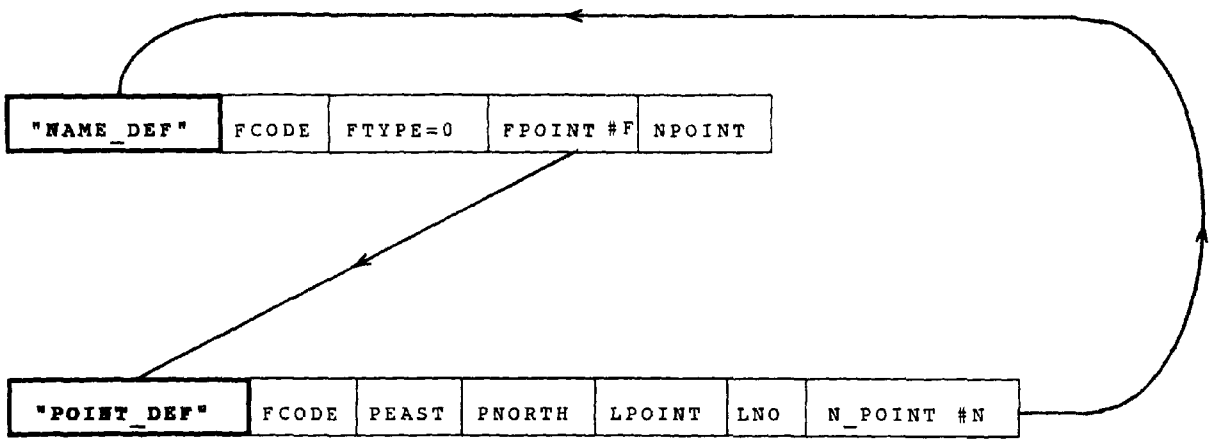


Fig 7.7 Extracting point data.

Table 7.6 "LINE_DEF" record structure

Field name (No)	Description
FCODE (1)	Feature Code
LLENGTH (2)	Line length (m)
PNODE1 (3)	Pointer to Node1
PNODE2 (4)	Pointer to Node2
LCOORDP (5)	Line coordinate pointer
LNOC (6)	No. of coordinate pairs
N_POINT (7)	Name pointer

Table 7.7 "LINE_COORD" record structure

Field name	Description
EAST	Coordinate pair Easting (m)
NORTH	Coordinate pair Northing (m)

Table 7.8 "LINK_NODE"

Field name	Description
NODE	Line pointer to file "LINE_DEF"

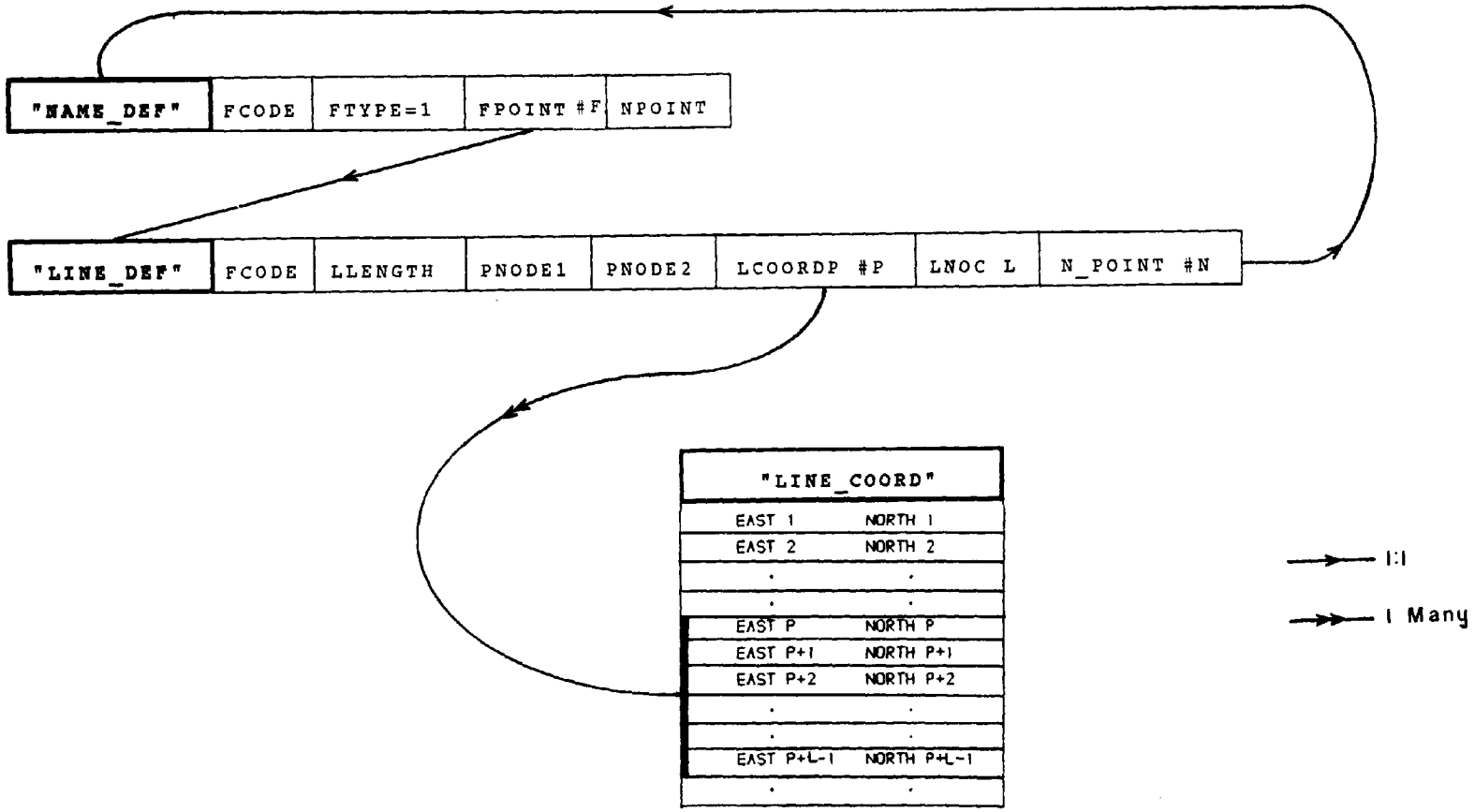
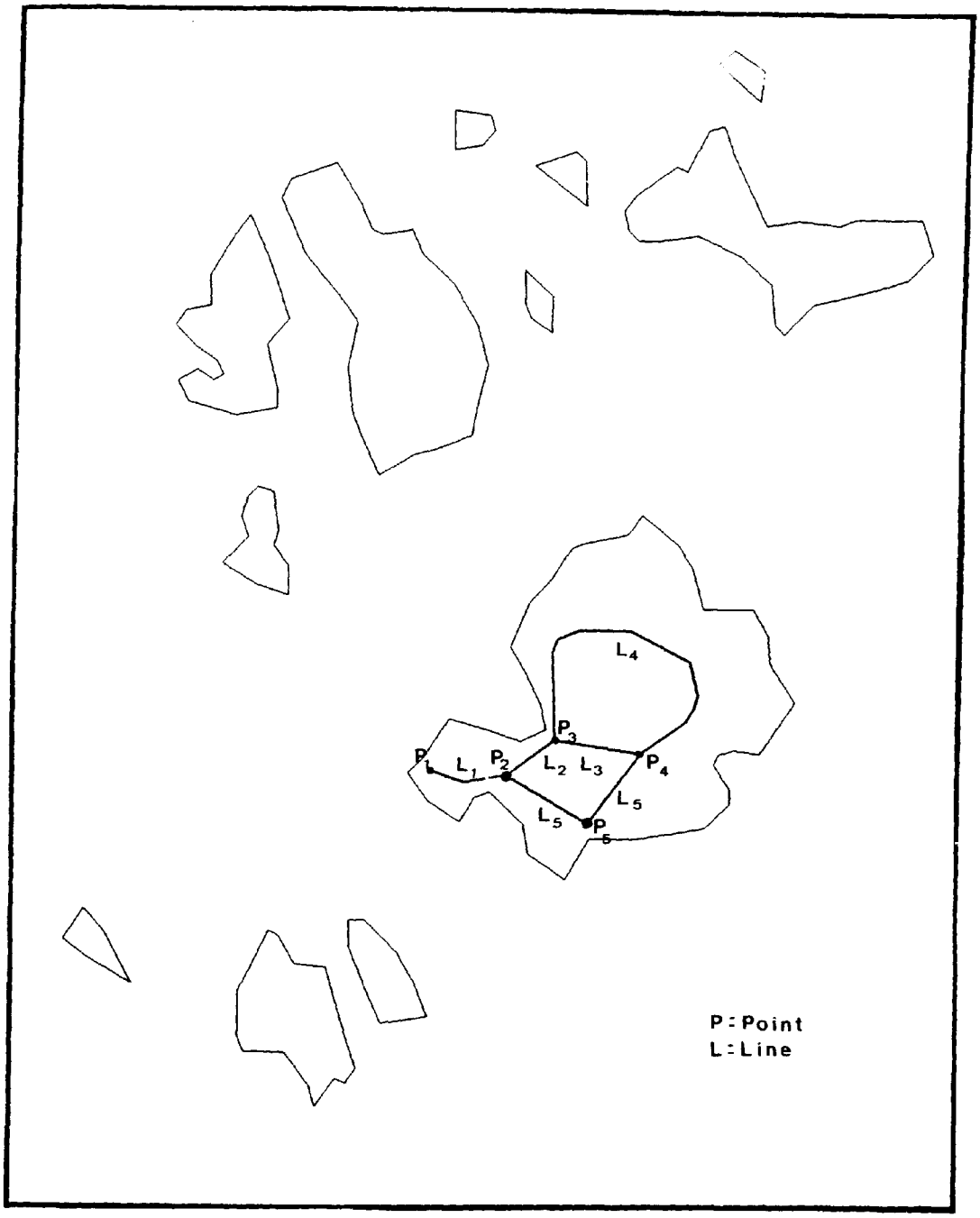


Fig 7.8 Extracting line data.



Record No.	"POINT_DEF" (NODE) file		"LINE_NODE" file	"LINE_DEF" file	
	LINE NODE Pointer	No. of records	POINT DEF Pointer	NODE 1 Pointer	NODE 2 Pointer
1	1	1	1	1	2
2	2	3	1	2	3
3	5	3	2	3	4
4	8	3	6	3	4
5	11	2	2	4	5
6	-	-	3	5	2
7	-	-	4	-	-
8	-	-	3	-	-
9	-	-	4	-	-
10	-	-	5	-	-
11	-	-	5	-	-
12	-	-	6	-	-

Fig 7.9 Node/line (road) network representation for St Mary's island in the Scilly Isles.

The coordinates making up lines are contained in the "LINE_COORD" file (Table 7.7). The contents of this are used for finding line label positions and can also be used for determining which sections of a line are straight enough to place a label on. The line coordinate data, can be indexed from the "LINE_DEF" file (Table 7.6) via the LCOORDP pointer and LNOC number of coordinate pairs fields.

It is often important to know whether a line is too short to be labelled, therefore information regarding the line length is required. To avoid having to calculate this by chaining along coordinate data held in the "LINE_COORD" file, the information is provided for in the "LINE_DEF" file.

7.4.4 AREA DATA

Areas are the only type of feature present, in the topological and feature classified data, which are stored in a rasterized form and hence all associated spatial attribute values are in pixel coordinates. Raster data rather than vector data is used because both the masking and area name placement primitives make use of this type of data.

"AREA_DEF" (Table 7.9) is used to store the main attributes about features of its type and has records indexed by the name index files. Several attributes required for name placement purposes are available. These can be used for deciding whether a label can fit into the area concerned. The elongation of an area is defined by dividing, the difference of the maximum and minimum moments of inertia by the sum (Winston and Horn, 1981). Alternatively the elongation could be defined in terms of the eccentricity of an ellipse fitted through the area.

A separate file "AREA_RAST" is used to store the raster data in run-length encoded form (Table 7.10) and is pointed to by RUN_POINT in "AREA_DEF". The run-length data structure, which is of the discontinuous form (Section 3.3.3 and Jones, 1987), is very useful for answering queries involving the testing of whether a point or a region lies inside a specific area feature. The run-length records, which are stored in order of increasing northings and eastings, can be quickly skipped over until either the location in question is found to be inside a run-length strip, or the maximum pixel northing of the specified location is passed, at which point the remainder of the run-length encoded data can be excluded from the search.

Table 7.9 "AREA_DEF" record structure

Field name (No)	Description
FCODE (1)	Feature Code
AAREA (2)	Area of area (pixels)
AORIENT (3)	Orientation of area (degrees)
AELONG (4)	Area elongation (0-1.0000)
ACM_EAST (5)	Area C/M Easting (pixels)
ACM_NORTH (6)	Area C/M Northing (pixels)
ASEED_EAST (7)	Area seed Easting (pixels)
ASEED_NORTH (8)	Area seed Northing (pixels)
MIN_EAST (9)	Minimum Easting (pixels)
MAX_EAST (10)	Maximum Easting (pixels)
MIN_NORTH (11)	Minimum Northing (pixels)
MAX_NORTH (12)	Maximum Northing (pixels)
RUN_POINT (13)	Run-length encoded raster pointer, points to coordinates in "AREA_RAST"
N_RUN (14)	Number of run-length strips
N_POINT (15)	"NAME_DEF" pointer

Table 7.10 "AREA_RAST" record structure

Field name	Description
R_EAST	Run-length Easting (pixels)
R_NORTH	Run-length Northing (pixels)
R_LENGTH	Run-length length (pixels)

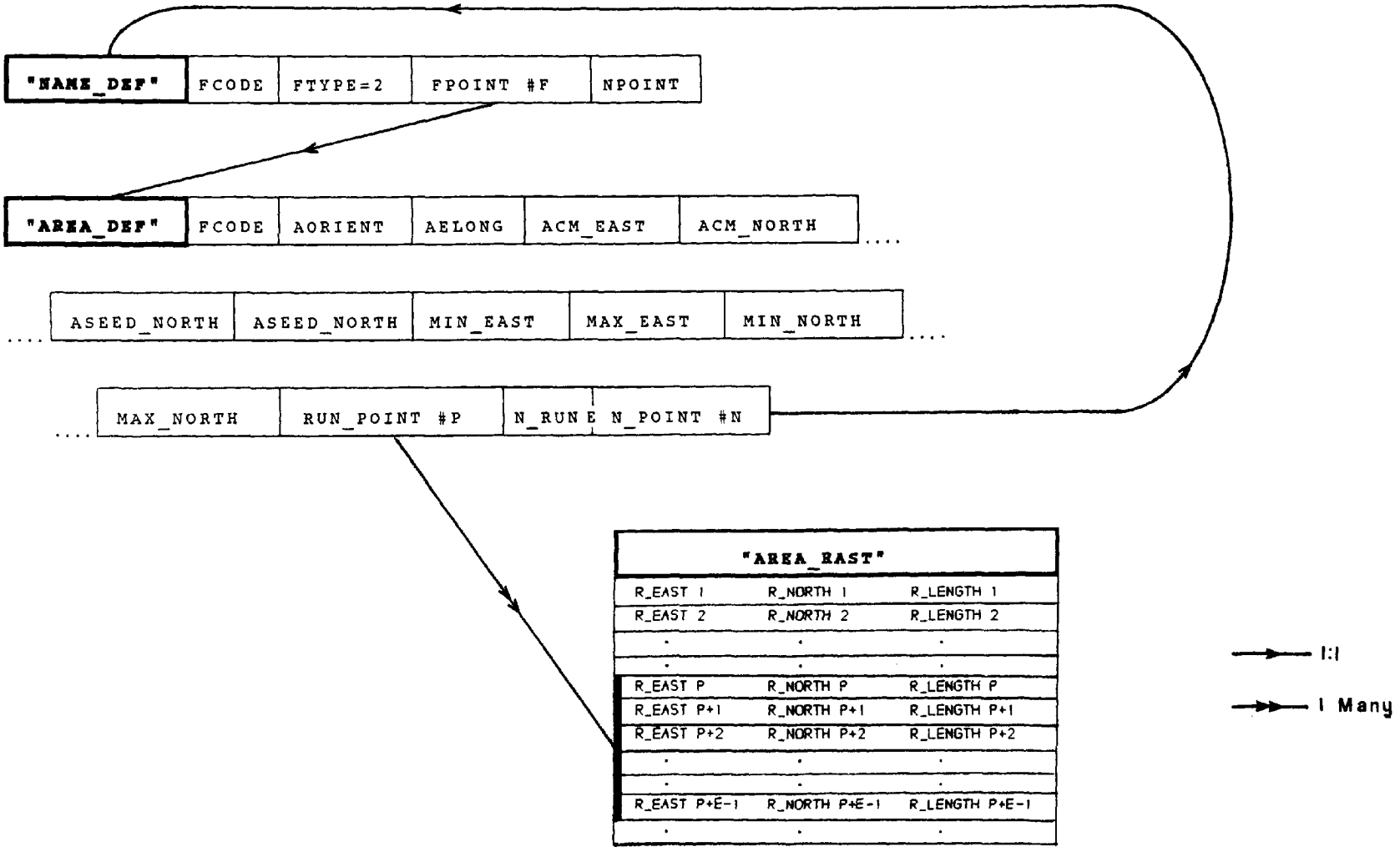


Fig 7.10 Extracting area data.

7.4.5 TOPOLOGICAL AND FEATURE DATA CLASSIFIED PRIMITIVES

Some PROLOG primitives are available for reading the fields of all topological and feature classified data files. The primitives for accessing the "POINT_DEF", "LINE_DEF" and "AREA_DEF" files are of the form:

get_point_....

get_line_....

get_area_....

For example, to access the easting coordinate of point Rec in the "POINT_DEF" file, the following primitive is used:

get_point_east(Rec,East).

Slightly higher level primitives are available for the extraction of related fields from files, such as:

get_point_coords(Rec,East,North).

Two exceptions to the general forms of topological and feature classified primitives, which extract feature codes and "NAME_DEF" pointers according to specified feature types, are:

get_feature_code(Ftype,Rec,Fcode).

get_feature_name_def_pointer(Ftype,Rec,Fcode).

Three general purpose primitives are available in order to retrieve whole lists of parameters from the associated files, by specifying a list of the field numbers:

read_point_details(Rec,Field_no_list,Parameter_list).

read_line_details(Rec,Field_no_list,Parameter_list).

read_area_details(Rec,Field_no_list,Parameter_list).

So for instance to extract the minimum and maximum eastings and northings (pixels) of an area, the following would be used:

read_area_details(N,[9,10,11,12],[Mine,Maxe,Minn,Maxn]).

The following two primitives:

get_line_coord_list(Line_no,Coords).

get_area_rast(Rec,East,North,Run).

are for extracting a list of coordinates making up a line and the lengths of individual area run-length strips.

Finally there are two validation primitives. The first one can be used to validate feature type and returns a type number (0,1 or 2). The second checks that a feature record is in range:

valid_ftype(Ftype,Type_no).

valid_feat(Ftype,Feat_rec).

A fuller description of the topological and feature classified data primitives can be found in Appendix 3.

7.4.6 SECTION CONCLUSION

The database structure provides enough topological point and line information to be of use in answering queries, such as listing all the named point locations along a main route. The topology of areas was not supported, but much of the required information can be obtained via the use of rasterized data on the map image and from run-length encoded data.

The cartographic data described here serves the purpose of uniquely identifying each feature, but does not give a global view of the map. Therefore, a raster map image must be used and this will be described in the next section.

7.5 RASTER MAP IMAGE

7.5.1 INTRODUCTION

The raster image of a map provides information in a pictorial form and allows certain spatial queries to be answered quickly. For instance, the amount of underlying detail in potential label positions, can be found simply by counting pixels within the region defined by the label. Also, the proximity of a label to the nearest feature may be determined by searching pixel locations away from the label, until a pixel is found containing a feature. Attempting to answer such queries using just cartographic vector data would involve a great deal of searching of the database and is therefore not practical.

The raster component of the database consists of the raster image, a map window definition file and a couple of raster image attribute files.

7.5.2 RASTER MAP IMAGE

The raster data structure involves the image being partitioned into small grid square files (Fig 7.11). This is akin to the library square concept of the Ordnance Survey 1:625000 scale Route Planner map database and the 7.5 minute quadrangles used by AUTONAP (Freeman, 1985), in

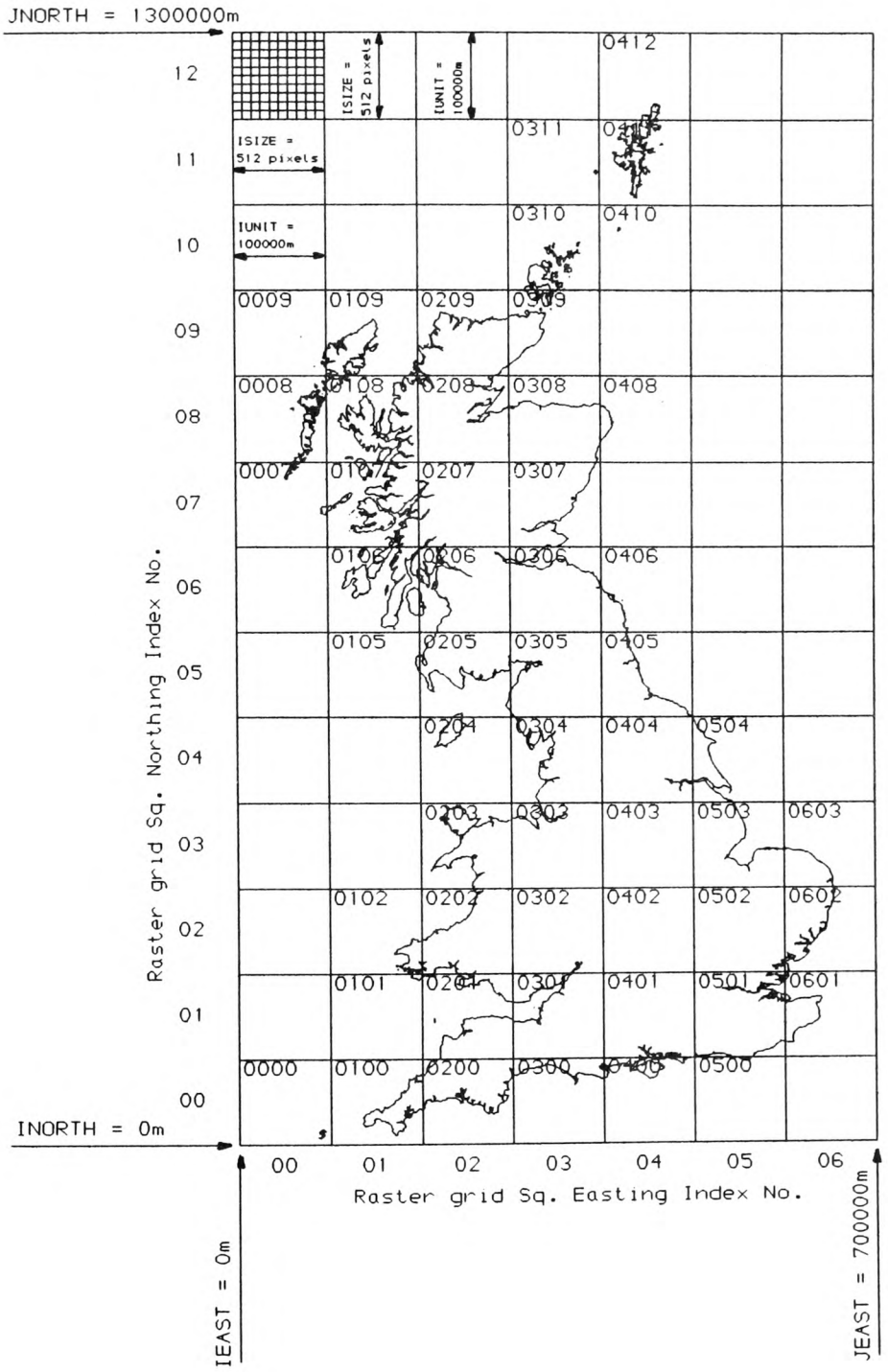


Fig 7.11 Raster grid square representation for the Route Planner map.

that to use just a small window region of the map, it should not be necessary to load the entire map. However in the current version of the NAMEX system this has not been implemented.

The reference system for the grid square image files, which was a four figure integer number, allowed from between 1 to 99 grid squares along each side of the map. The first two digits correspond to the grid square easting number and the second two digits refer to its northing number. Each grid square file was prefixed by an "R" to indicate that it is a run-length encoded file.

The raster data for each grid square is compacted into run-length encoded form, similar to row order (Section 3.3.3), and then saved as an unformatted binary file to reduce storage requirements during the database generation.

7.5.3 MAP WINDOW

The "WIND_DEF" file (Table 7.11), contains information related to map and raster image window specifications as well as the raster image scale, (pixels per grid unit). It is used by the DB_GENERATE program to define the map region covered and by DB_ACCESS for map image re-constitution. Because the ability to "window-in"

on the generated raster image was not implemented (Section 7.5.2), the map and raster window limits are by default the same.

Fig 7.11 illustrates how the Route Planner map of Great Britain could be divided up into grid squares. However, in practice it is not possible to place all the names on the Route Planner map at once using NAMEX due to search space memory limitations in POPLOG.

7.5.4 RASTER AND FEATURE DEFINITION

The "FEAT_DEF" file contains attribute data pertaining to each feature code. The attributes consist of a written description of the coded feature class, its width in metres on the ground, and the raster bit plane it resides in. If a feature of a particular code is not to be rasterized, then its width (not applicable to area features) and bit plane are set to zero. The feature name is written in lower case and separate words joined by underscores so that these can be easily utilised in PROLOG.

The "RAST_DEF" file gives a class description of each bit plane in the rasterized image and the associated bit plane priority. By default, bit plane zero is defined as the "rub_out" raster plane and bit planes 1 to 29

inclusive can be assigned any class description. As adopted in LABPOS, the higher the priority value, the more important a feature class and the less permissible it is to place a name over such a feature. The priority of each bit plane can be adjusted during the name placement process if necessary. As with the "FEAT_DEF" file, the class name of the feature is written in lower case letters with underscores between the words. Both files can be edited via DB_DEF called from option five in the NAMEX menu.

7.5.5 RASTER IMAGE PRIMITIVES

There are three types of raster image data primitives:

i. The analysis of specified window regions in the image and storage of the sum of pixel contents in an array in DB_ACCESS.

ii. The retrieval of pixel contents from DB_ACCESS and the analysis of these in PROLOG.

iii. The ability to mask or rub-out (Section 6.5.3) labelled features so that when examining underlying detail at possible placement positions the labelled feature is ignored.

The window regions considered for examining the raster image are either point locations, rectangular or circular. All coordinates and dimensions are in metres and angles are in degrees:

rast_point_init(X,Y).

rast_circle_init(X,Y,Radius).

rast_rect_init(X,Y,Length,Width,Angle).

Analysis of the raster data involves examining the sum of the pixels in each bit plane in the raster window boundary. This information can be retrieved by calling a primitive which returns the value of the pixel sum found in the specified bit plane:

rast_plane_read(Plane,Value).

The number of pixels beyond the map edge, the total number of pixels examined and the total number of unoccupied pixels in the window region can also be found:

rast_over_edge(No).

rast_total_pix(No).

rast_free_pix(No).

To give an overall impression of suitability of the pixel contents within a window region, the sum of all feature occupied pixels present in bit planes multiplied by their associated priority can be found by calling:

window_pixel_sum(Pixel_sum).

When examining the suitability of label positions with respect to underlying features, it is necessary to ignore the presence of the feature that the label represents. To mask out a feature of serial number Fsn and type Ftype on a map image the following primitive is used:

mask_out_feature(Fsn,Ftype).

A variant of this primitive can be used to temporarily mask out the feature belonging to the current label:

mask_out_current_feature.

Both of the above masking primitives automatically remove the presence of any previous mask before overlaying the new mask. If however it is desired to erase the current mask then the following primitive is called:

erase_mask.

Finally, to load the raster map image, the primitive:

load_raster_image.

must be called. This additionally copies data contained in the map window (Table 7.11), feature (Table 7.12) and raster definition (Table 7.13) files into the NAMEX system in the form of PROLOG facts:

map_coords([bottom_left,East,North]).

map_coords([top_right,East,North]).

Table 7.11 "WIND_DEF" file record contents

Record	Name	Description
1	IEAST	Easting of bottom left corner on map (m).
2	INORTH	Northing of bottom left corner on map (m).
3	JEAST	Easting of top right corner on map (m).
4	JNORTH	Northing of top right corner on map (m).
5	ISIZE	Pixel size of map grid sq. unit (pixels).
6	IUNIT	Map grid sq. size (m).
7	LEAST	Easting of bottom left corner of image (m).
8	LNORTH	Northing of bottom left corner of image (m).
9	MEAST	Easting of top right corner of image (m)
10	MNORTH	Northing of top right corner of image (m).
11	ISCALE	Map scale.

Table 7.12 "FEAT_DEF" record structure

Field name	Description
FCODE	Feature code (0-999).
FEAT	Feature name (lower case letters).
WIDTH	Width (0-999999m).
PLANE	Bit plane (0-30).

Table 7.13 "RAST_DEF" record structure

Field Name	Description
PLANE	Bit Plane (0-30).
CLASS	Feature Class (lower case).
PRIOR	Priority (-999 to 9999).

```
rast_scale([Isize,Iunit]).

scale(Sca).

feat_def(Code,Feature,Width,Plane).

rast_def(Plane,Feature_class,Priority).
```

7.5.6 SPATIAL QUERIES USING THE RASTER PREDICATE (PRIMITIVES)

The raster data complements the topological and feature classified data, previously used to establish a set of possible positions for each label, by allowing an assessment of the quality of these with respect to underlying features.

The few raster primitives discussed provide the "eyes" of the NAMEX system with which spatial information can be utilised to answer questions such as "is town A near enough to the coast to place the label on the seaward side of the coastline?" (rule [2.22]). Given that the town is at coordinates E,N and that "near" means within 2km or 2000m, then rule [2.22] is satisfied by:

```
rule2_22(E,N):-
    /* Examine pixel contents within 2km radius */
    rast_circle_init(E,N,2000),
    rast_def(Plane,coast,Priority), /* Get bit plane */
    rast_plane_read(Plane,Value), */ No. for coast */
    Value > 0. /* Any pixels in coast bit plane ? */
```

More elaborate primitives can be constructed from the basic raster primitives described. For instance it is possible to determine if a city label is placed on the wrong side of a county boundary to the city (Section 8.4.6.2).

The next section describes the label status register and parameterised rule files which define which positions, configurations and sizes are available for labels.

7.6 LABEL STATUS REGISTER AND PARAMETERIZED TEXT DEFINITION RULE-BASE

7.6.1 INTRODUCTION

This section of the chapter describes the label status register, which keeps a record of the current position, size and configuration of each label. Because features are classified according to their feature code attribute, labels are classified similarly.

Also described here is a rule-base which is in the form of several parameterized text definition files which define the set of configurations, sizes and positions permissible for labels of different feature codes. It also includes a font definition file which is used to specify the character font sizes available to the system. Although the text definition rule-base allows for most of the recommended label positions and configurations discussed in chapter 5, its use is optional since the user may wish to define his own label positions and configurations in the LOGIC program.

7.6.2 LABEL STATUS REGISTER

This consists of three files, "LABEL", "MULTI_WORD" and "CURVED_LABEL". The "LABEL" file describes the status

of simple rectangular labels, which are the most common form of label (Table 7.14). However, on some occasions, the words or letters in a label can be widely separated (Fig 7.12). Although these label configurations are not implemented in the current version of NAMEX, two files were designed to cater for these: "MULTI_WORD" (Table 7.15) and "CURVED_LABEL" (Table 7.16). The relationship between all three files and the "LABEL" file can be seen in Figs 7.13-7.15.

In the label status register files, only the fields associated with the current label configuration contain parameters, the remainder are left blank. An example of a typical label status field in the "LABEL" file is L_CONFIG which is used to give a unique descriptive number to the label's configuration (Fig 7.18). Another example label status field is L_NAME which contains the current version of the name. This is useful because in high feature density areas of a map it is sometimes permissible to abbreviate the name. The original version of the name can of course be accessed from the "NAME" file. Further information, regarding the contents of the label status register files, can be found in tables 7.14-7.16 and in the separate program documentation.

Table 7.14 "LABEL" record structure

Field name (No)	Description
L_TSN (1)	Pointer to "NAME".
L_NSN (2)	Pointer to "NAME_DEF".
L_FSN (3)	Pointer to feature.
L_TYPE (4)	Feature type (0,1,2).
L_LENGTH (5)	Label length (m).
L_PROX (6)	Label proximity to feature (m).
L_ARROW (7)	Label arrow distance (m).
L_POS (8)	Label position No.
L_EAST (9)	Label easting (m).
L_NORTH (10)	Label northing (m).
L_ANGLE (11)	Label angle (degrees).
L_FONT (12)	Label font No.
L_LSPACE (13)	Label letter spacing (m)
L_WSPACE (14)	Label word spacing (m)
L_LISPACE (15)	Label line spacing (m)
L_NAME (16)	Label name (abbreviated ?)
L_NC (17)	Label No. of alphanumeric characters.
L_SPLIT (18)	Label No. of times split.
L_JUST (19)	Label justify.
L_POINTER (20)	Pointer to extra attributes.
L_NO (21)	No. of extra attributes.
L_CONFIG (22)	Label configuration. *
AREA_P1 (23)	Area parameter 1 e.g. Ax^3.
AREA_P2 (24)	Area parameter 2 e.g. Bx^2.
AREA_P3 (25)	Area parameter 3 e.g. Cx.
AREA_P4 (26)	Area parameter 4 e.g. D.

* Key to label configuration (Fig 7.18)

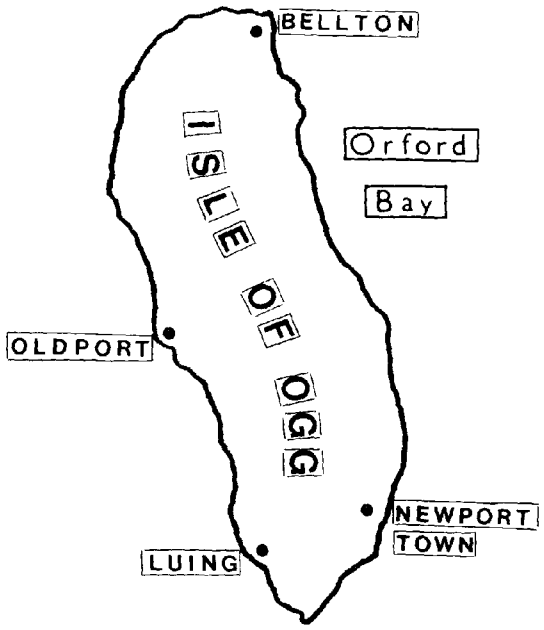
0 - horizontal configuration.
1 - diagonal configuration.
2 - circular arc configuration.
3 - parabolic configuration.
4 - "S" shaped or cubic configuration.
5 - horizontal line name, offset to the left.
6 - horizontal line name, offset slightly to the left.
7 - horizontal line name centred on the line.
8 - horizontal line name, offset slightly to the right.
9 - horizontal line name, offset to the right.
10 - upside down names permitted.

Table 7.15 "MULTI_WORD" record structure

Field name	Description
W_NAME	Word name.
W_NO	Word No. of letters.
W_EAST	Word easting (m).
W_NORTH	Word northing (m).
W_ANGLE	Word angle (degrees).
W_LENGTH	Word length (m).
W_POINTER	Curved label pointer.

Table 7.16 "CURVED_LABEL" record structure

Field name	Description
L_EAST	Letter easting (m).
L_NORTH	Letter northing (m).
L_ANGLE	Letter angle (degrees).



"LABEL" "MULTI_WORD" "CURVED_LABEL"

OLDPORT

LUING

NEWPORT TOWN — NEWPORT, TOWN

BELLTON

Orford Bay — Orford, Bay

OLDPORT

ISLE OF OGG — ISLE (I S L E), OF (O F), OGG (O G G)

Fig 7.12 Graphical representation of single, multi-worded and curved labels and associated file structure.

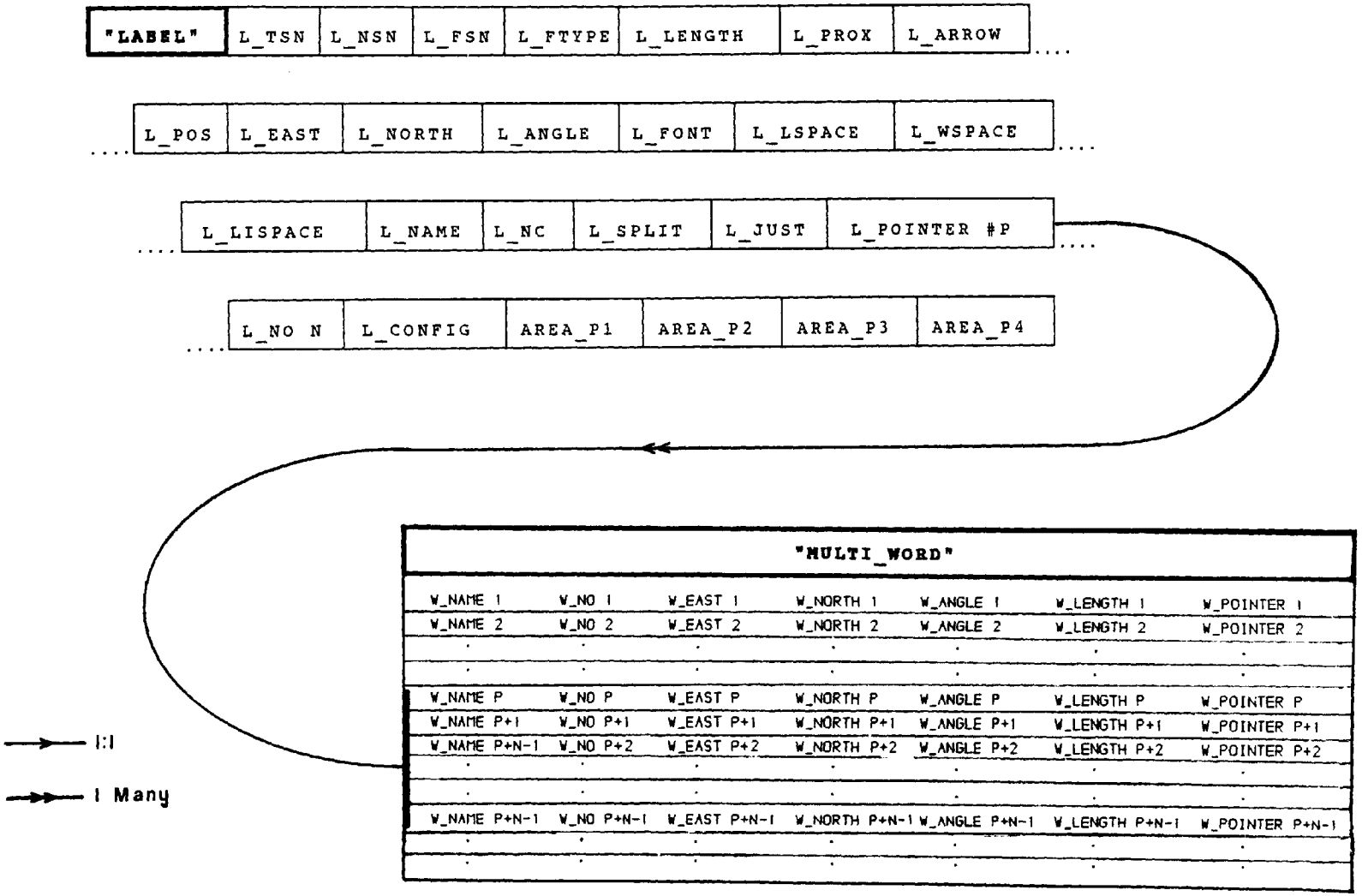


Fig 7.13 Extracting multiply-worded labels using "LABEL".

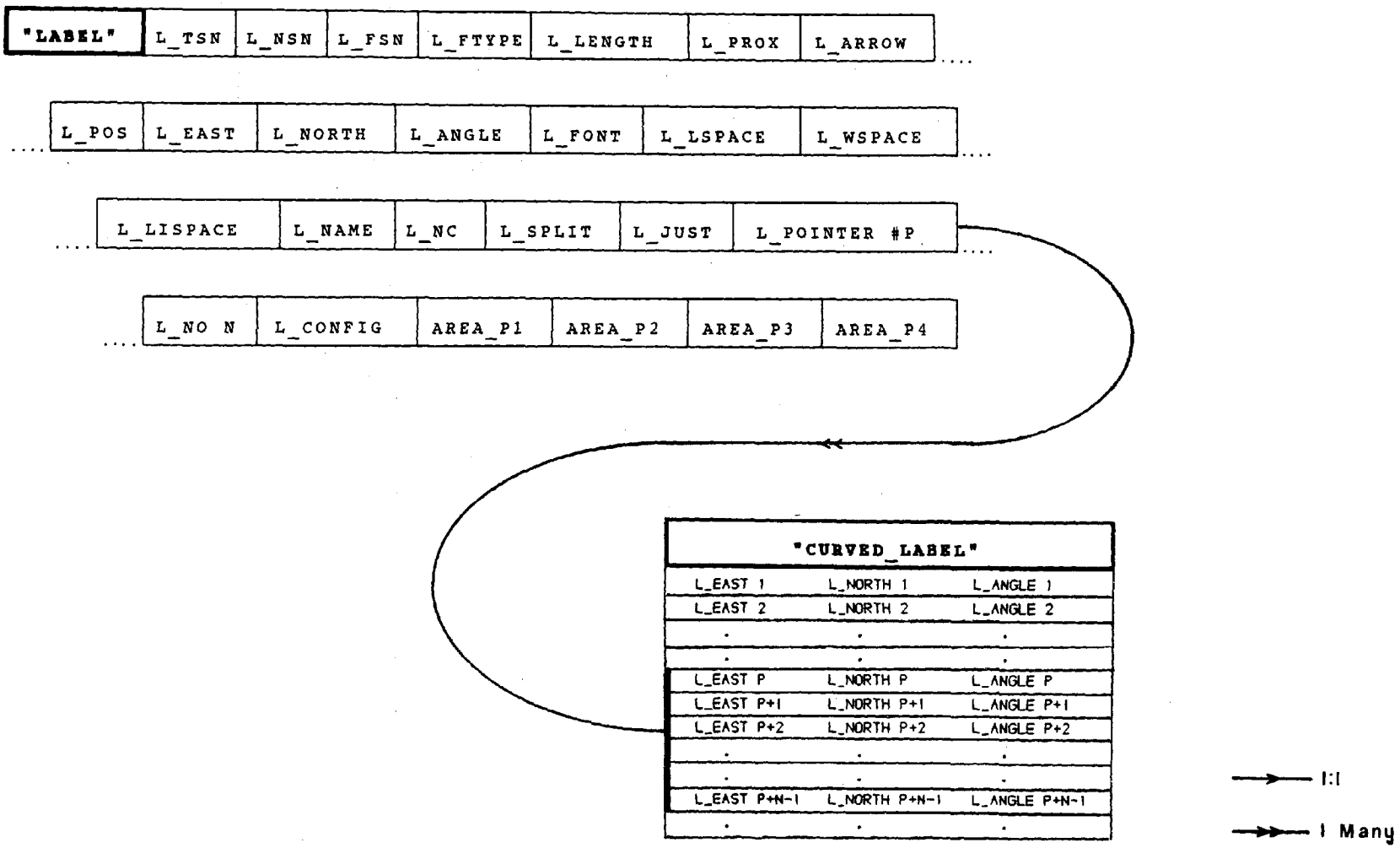


Fig 7.14 Extraction of the positions of individual letters in curved labels using the "LABEL" file.

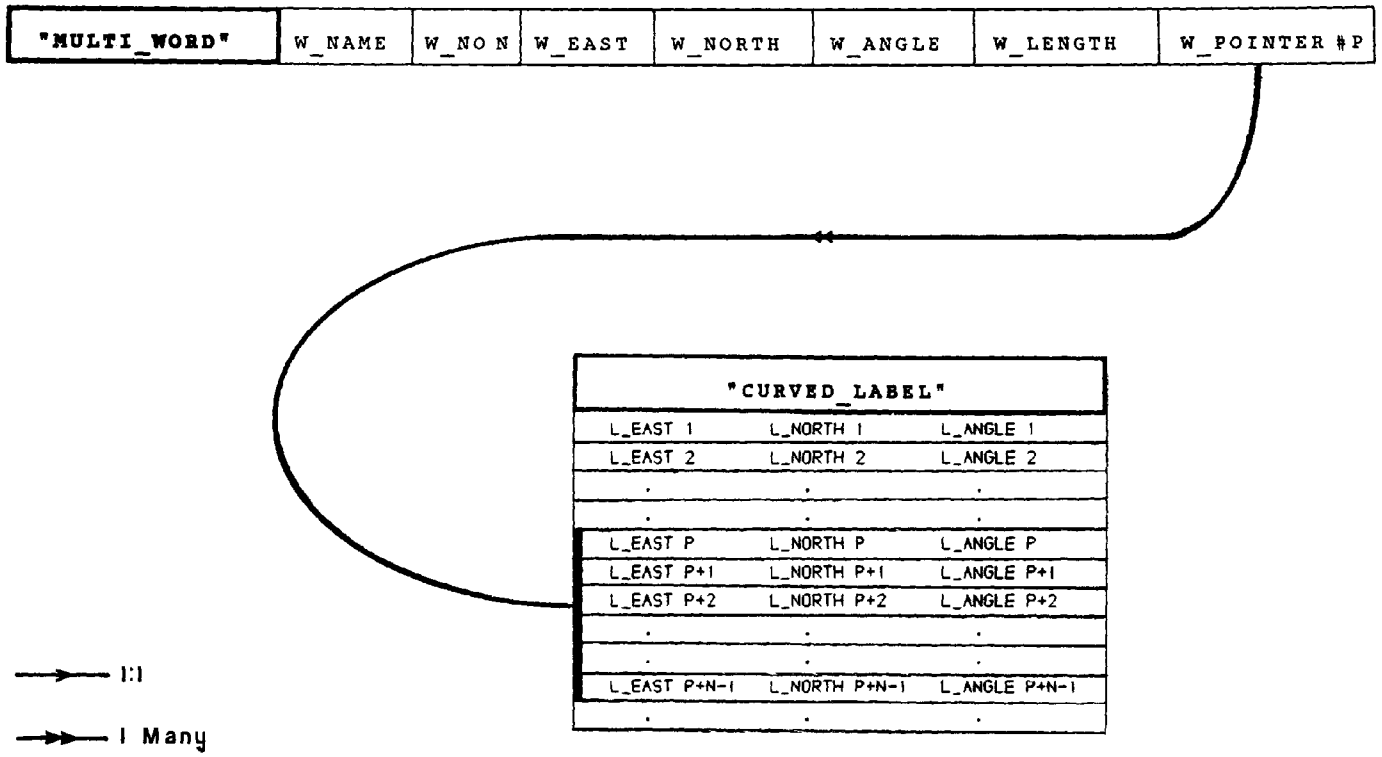


Fig 7.15 Extraction of the positions of individual letters in words contained in the "MULTI_WORD" files.

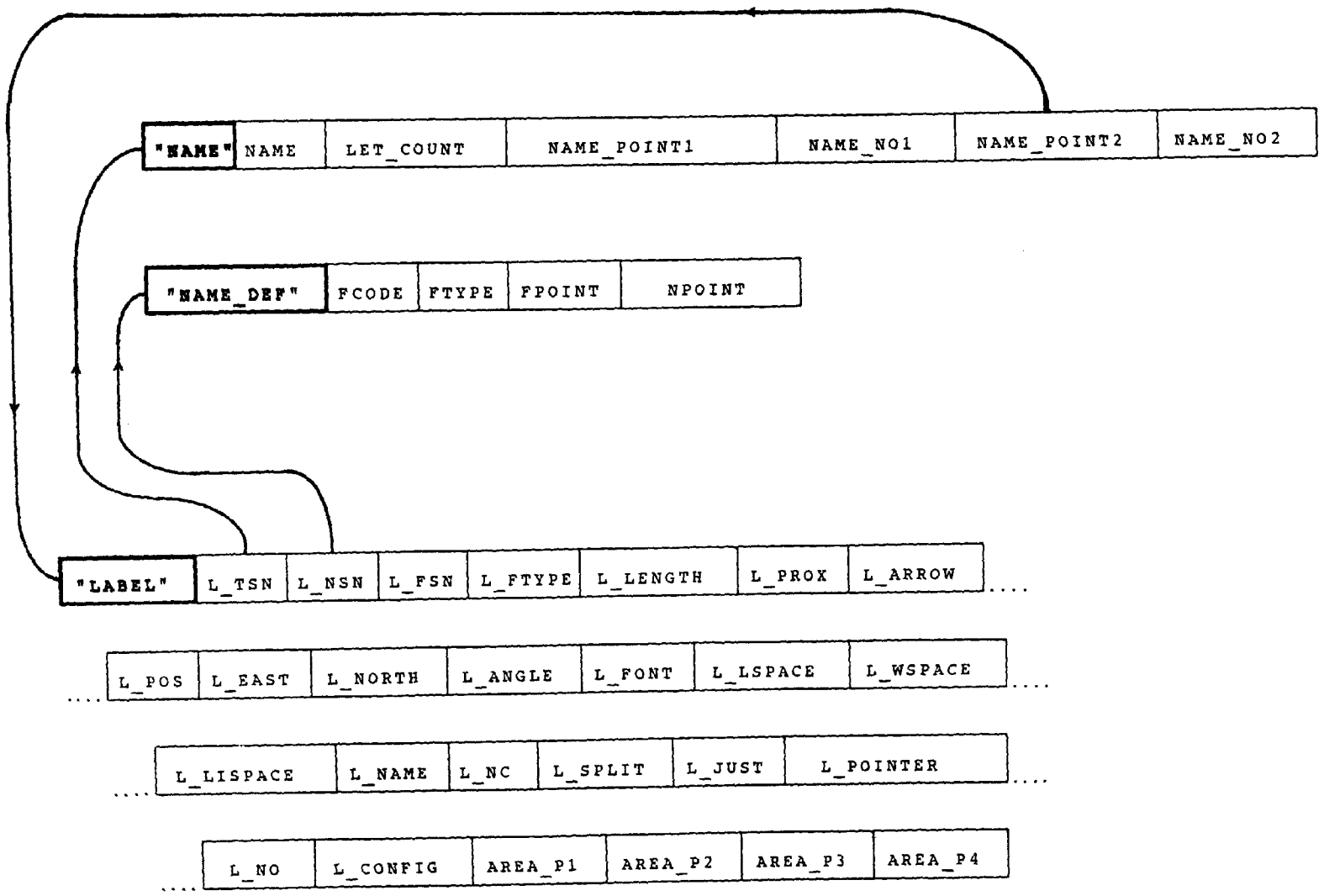


Fig 7.16 Extracting name index files using "LABEL".

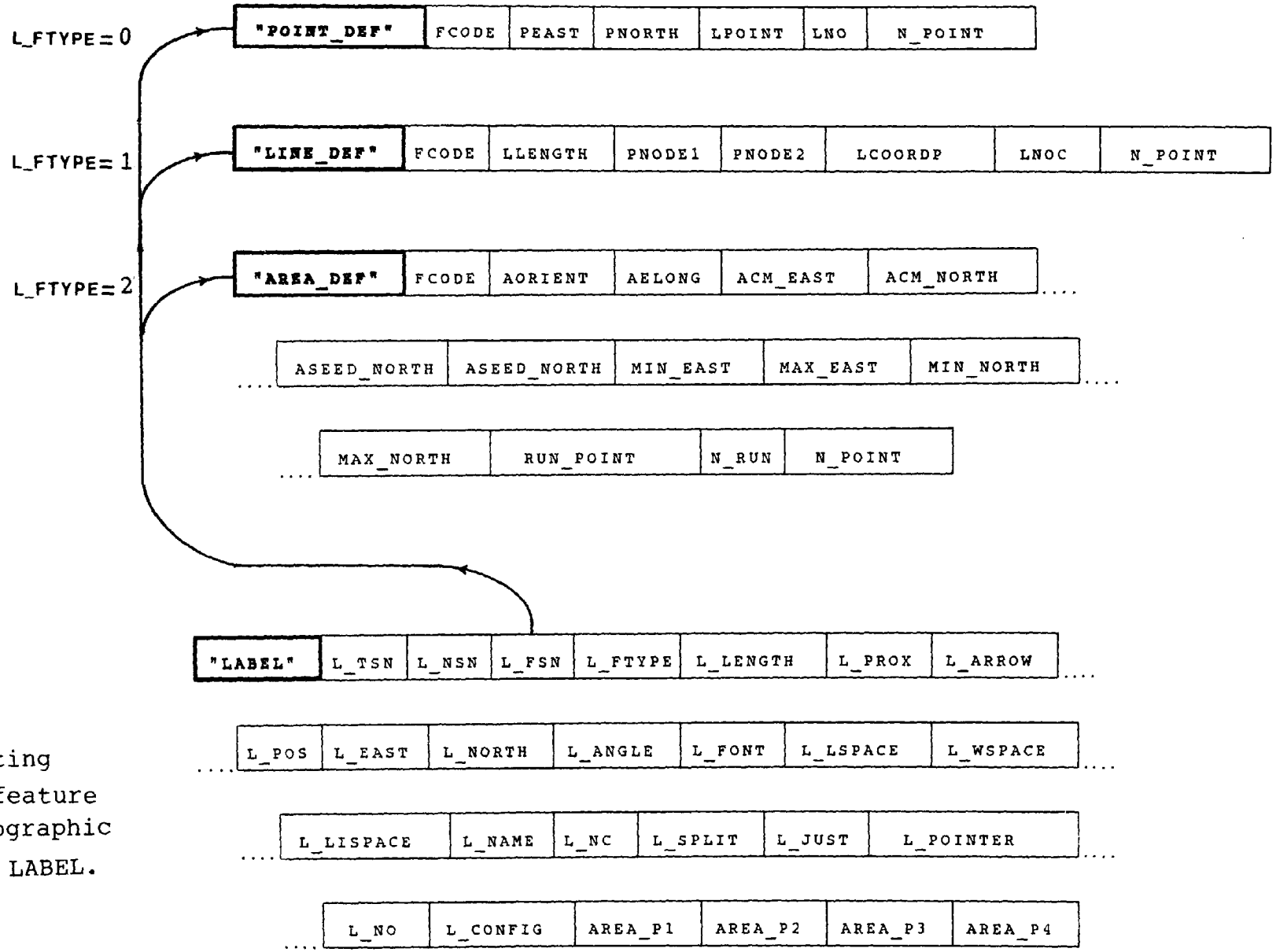


Fig 7.17 Extracting topological and feature classified cartographic data files using LABEL.

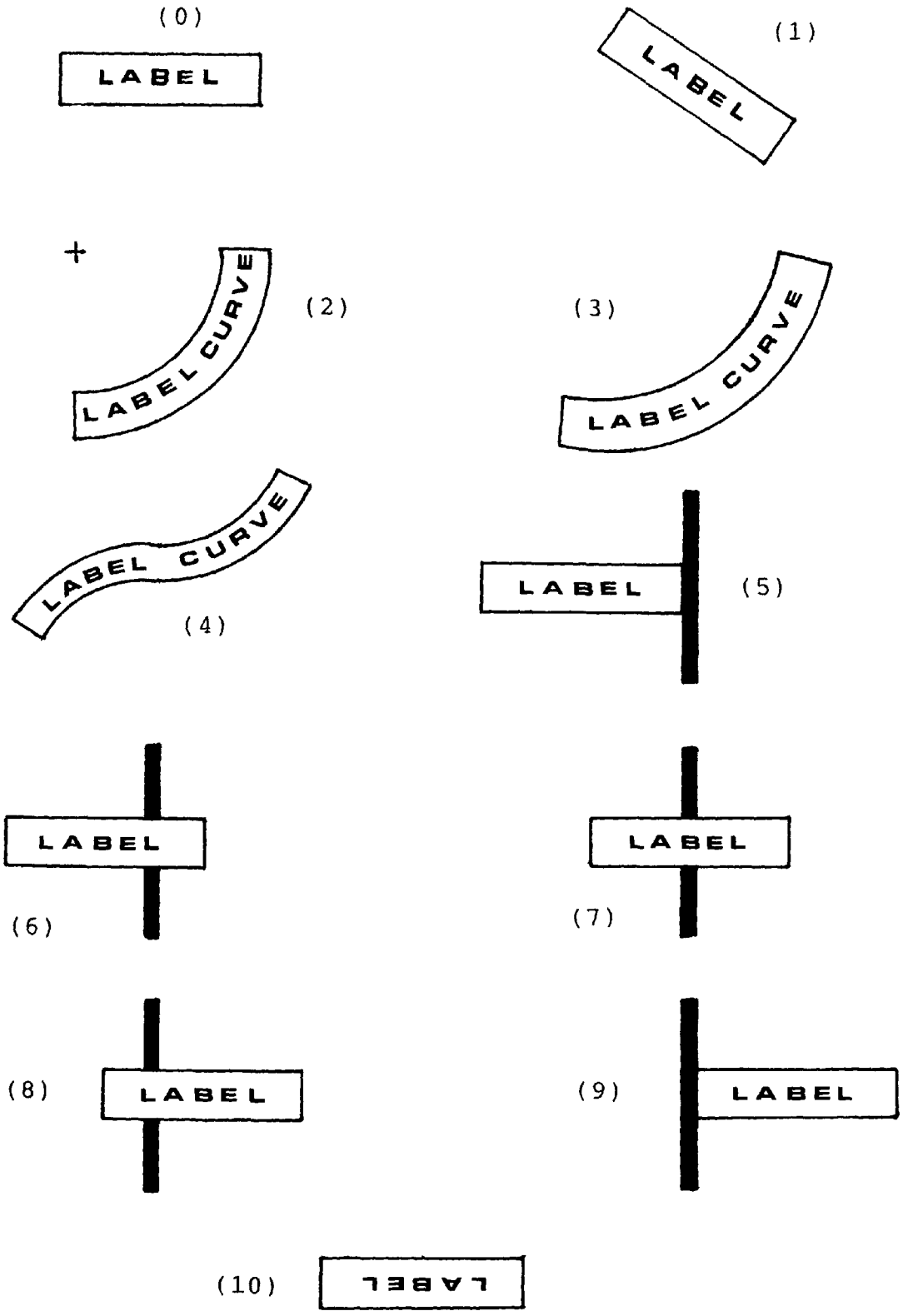


Fig 7.18 Label configuration classes.

7.6.3 PARAMETERIZED TEXT DEFINITION RULE-BASE

Text definition files are intended to be generated by an experienced cartographer, via DB_DEF (NAMEX menu option five) and form a parameterized rule-base which can be called upon from NAMEX to aid the selection of label sizes, positions and configurations used. The rules in the rule-base generally take the form of preference of use (%) for certain label positions and configurations, and minimum, typical and maximum values for other attributes such as font size.

The parameterized text definition rule-base consists of three files. Each file allows for a thousand feature codes, however most of these feature codes are unlikely to be used and so their corresponding records are left blank. The recommended configurations which are common to most types of labels (Chapter 5) are catered for (however several of these have not been implemented in the current version of NAMEX) in the "TEXT_PARAM" file, the record structure for which is given in table 7.17. Two extra files, "TEXT_LINE" (Table 7.18) and "TEXT_AREA" (Table 7.19), were intended to define additional configurations for labels of line and area types but are not implemented in the current version of NAMEX. Further information regarding the parameterized text definition files can be found in separate documentation.

Table 7.17 "TEXT_PARAM" record structure

Field name (No)	Description
PROX (1)	Typical proximity distance (m).
PROX_MIN (2)	Minimum proximity distance (m).
ARRW_DIST (3)	Maximum arrow distance (m).
POS(1) (4)	Position 1 (%).
POS(2) (4)	Position 2 (%).
POS(3) (4)	Position 3 (%).
POS(4) (4)	Position 4 (%).
POS(5) (4)	Position 5 (%).
POS(6) (4)	Position 6 (%).
POS(7) (4)	Position 7 (%).
POS(8) (4)	Position 8 (%).
POS(9) (4)	Position 9 (%).
POS(10) (4)	Position 10 (%).
POS(11) (4)	Position 11 (%).
POS(12) (4)	Position 12 (%).
POS(13) (4)	Position 13 (%).
POS(14) (4)	Position 14 (%).
POS(15) (4)	Position 15 (%).
POS(16) (4)	Position 16 (%).
POS(17) (4)	Position 17 (%).
POS(18) (4)	Position 18 (%).
POS(19) (4)	Position 19 (%).
POS(20) (4)	Position 20 (%).
FONT_MIN (5)	Minimum font No.
FONT (6)	Typical font No.
FONT_MAX (7)	Maximum font No.
LT_MIN_SP (8)	Minimum letter spacing (m).
LT_SP (9)	Typical letter spacing (m).
LT_MAX_SP (10)	Maximum letter spacing (m).
WD_MIN_SP (11)	Minimum word spacing (m).
WD_SP (12)	Typical word spacing (m).
WD_MAX_SP (13)	Maximum word spacing (m).
LN_MIN_SP (14)	Minimum line spacing (m).
LN_SP (15)	Typical line spacing (m).
LN_MAX_SP (16)	Maximum line spacing (m).
ABRV (17)	Usage of abbreviation (%).
SPLIT_0 (18)	Not split (%).
SPLIT_1 (19)	Split once (%).
SPLIT_2 (20)	Split twice (%).
SPLIT_3 (21)	Split thrice (%).
L_JUST (22)	Left justication (%).
C_JUST (23)	Centre justification (%).
R_JUST (24)	Right justification (%).
HOR_PLAC (25)	Horizontal placement (%).
DIG_PLAC (26)	Diagonal placement (%).
CUR_PLAC (27)	Curved placement (%).
ARW_PLAC (28)	Arrowed placement (%).
REP_MIN_DIST (29)	Minimum repeat distance (m).
REP_DIST (30)	Typical repeat distance (m).
REP_MAX_DIST (31)	Maximum repeat distance (m).
F_TYPE (32)	Feature type (0,1 or 2).

Table 7.18 "TEXT_LINE" record structure

Field name	Description
OF_ND_MIN	Minimum offset node distance (m).
OF_ND	Typical offset node distance (m).
OF_ND_MAX	Maximum offset node distance (m).
TP_LIN	On top of line (%).
CN_LIN	Centred on line (%).
BT_LIN	On bottom of line (%).
HOR_LFT	Horizontal left (%).
HOR_M_LFT	Horizontal mid left (%).
HOR_CEN	Horizontal centre (%).
HOR_M_RHT	Horizontal mid right (%).
HOR_RHT	Horizontal right (%).
UPSD	Upside down (%).

Table 7.19 "TEXT_AREA" record structure

Field name	Description
ARC	Arc shaped (%).
PARABOLA	Parabola shaped (%).
S_SHAPE	"S" shaped (%).
I_AREA	Inside area (%).
O_AREA	Outside area (%).
I_O_AREA	In and out of area (%).
MIN_OF_LF	Minimum left offset (m).
MIN_OF_RT	Minimum right offset (m).
MIN_OF_UP	Minimum upper offset (m).
MIN_OF_LW	Minimum lower offset (m).

Table 7.20 "FONT" record structure

Field name	Description
BL_HT	Letter block height (m).
BL_WT	Letter block width (m).
LT_HT	Letter height (m).
LT_WT	Letter width (m).
LC_HT	Lower case height (m).
LD	Letter descender (m).
SP1	Spare1.
SP2	Spare2.
SP3	Spare3.

7.6.4 FONT DEFINITION

The use of different character fonts is catered for in the NAMEX system by the "FONT" file. This defines the graphical characteristics of different fonts in terms of metres on the ground. It has three spare fields to allow for additional descriptors such as colour or line thickness, however these are not accessible in the current system. A full description of the font parameterization is given in table 7.20 (Also see Fig 5.1 and Section 6.4.4.1).

7.6.5 STATUS REGISTER AND TEXT DEFINITION RULE-BASE PRIMITIVES

Because of the large numbers of fields involved with each of the label status register and text definition files, access to field information is made by passing lists of parameters. For instance:

read_label_details(Rec,Field_list,Parameter_list).

reads the contents of a particular record in the "LABEL" file and passes back the required field contents in a list. The Field_list consists of a list of field numbers that the user wishes to access. The Parameter_list

contains a list of variables corresponding to and in the same order as the fields in the Field_list.

write_label_details(Rec,Field_list,Parameter_list).

can be used to write a list of parameters to a record. When passing text, one must pass the number of letters prior to the name.

valid_label(Rec).

is used to validate label Rec as being suitable to place according to criteria defined in the "name_select" predicate. The "name_select" predicate is defined in the LOGIC program, see section 7.8.6 for further details.

The "TEXT_PARAM" text definition rule-base file can make use of a parameter list for reading fields, but because access to the "TEXT_PARAM" file can include sub-field numbers, an alternative means of access is provided whereby field contents are accessed one at a time. This is particularly useful when accessing each of the twenty label position preferences where the first parameter, Param1, corresponds to the field number and the second, Param2, corresponds to the position.

read_text_param(Fcode,Param_list,Variable_list).

read_text_param(Fcode,Param1,Param2,Value_out).

write_text_param(Fcode,Param1,Param2,Value_in).

are used to read and write to the contents of the text definition rule-base files. The contents of these files are stored in FORTRAN memory during the operation of NAMEX and are initially loaded with a call to:

read_text_param.

If any of the fields have been changed, then these can be saved as an updated version to file:

write_text_param.

In the selection of features to label, it is possible to choose these on a feature code basis prior to some additional selection criteria later. To ban specific features from being labelled, these can be given a font number of zero in the "TEXT_PARAM" file. Because of the importance of a feature's font number, two font primitives are provided:

get_text_param_font(Fcode,Font).

valid_font(Font).

The former accesses the "TEXT_PARAM" file directly for the font number without having to use a list of parameters. The latter performs a check on whether a feature is valid to be labelled by testing to see if the given font number is non-zero.

Finally:

output_label_count(No).

can be called to find out how many labels have been written to.

7.6.6 SECTION SUMMARY

This section has described a very flexible, but complicated label status register. It is complicated by the fact that there are very large numbers of label configurations which can be selected from the parameterized rule-base. However, the selection of label configurations is controlled from the LOGIC program and if necessary any or all of the recommended label configurations can be overruled.

The next section describes the relatively simple file structure of the label output data.

7.7 NAME PLACEMENT OUTPUT

The design of the name placement output data must allow for a transfer format suitable for easy graphical output. Storing the positions and orientations of each individual letter in a sequential file, is a practical solution. It enables labels with letters, either closely spaced together or widely separated, to be rapidly plotted using standard graphics packages such as GINO-F or GKS. The output file, "LABEL_OUTPUT", contains mixed length records whose format are presented in table 7.21. High level pseudo code for graphical output is presented below:

```
BEGIN
  plot map
  open label output file
  WHILE NOT(EOF)
     read label header
     (* get information on Font & No. of characters *)
     select font
     FOR all label characters
        read character coordinates & angle of tilt
        plot character
     ENDFOR
  ENDWHILE
  close label output file
END
```

Table 7.21 "LABEL_OUTPUT" record structure

Field name	Description
FSN	Feature Serial No.
ND_POINT	"NAME_DEF" name pointer
N_POINT	"NAME" name pointer
NO_LET	No of letters (N)
NAME	Name
FONT_NO	Font No
Followed by records corresponding to each letter:	
E(1)	Easting (m) of letter 1 \
N(1)	Northing (m) of letter 1 Rec 1
A(1)	Orientation (degrees) of letter 1 /
E(2)	Easting (m) of letter 2 \
N(2)	Northing (m) of letter 2 Rec 2
A(2)	Orientation (degrees) of letter 2 /
E(3)	Easting (m) of letter 3 \
N(3)	Northing (m) of letter 3 Rec 3
A(3)	Orientation (degrees) of letter 3 /
.	. .
.	. .
.	. .
E(N)	Easting (m) of letter N \
N(N)	Northing (m) of letter N Rec N
A(N)	Orientation (degrees) of letter N /

Selection of character font specifications and sizes can be obtained by using the font field from "LABEL_OUTPUT" to point to font dimensions retained in the "FONT" file. Because the file is designed for the quick output of letters, all space characters have been removed from the name field.

Option four of the NAMEX user menu activates the writing of label records to the output file. If any labels have been flagged as removed then these will be excluded from the label output file. The following primitives are used to open, the "LABEL_OUTPUT" file, output to it and to close it:

open_label_output.

output_label(Rec).

close_label_output.

The next section of the chapter describes the name placement primitives.

7.8 NAME PLACEMENT PRIMITIVES

7.8.1 INTRODUCTION

Surprisingly there are relatively few name placement primitives compared to the number of primitives for querying the database. This is because acquiring information relevant to deciding where to place a label is more complicated than the process of actually placing a label itself. Five types of name placement primitives exist:

i. Computation of label dimensions.

ii. Computation of label position.

iii. Determination of labels in potential overlap.

iv. Label conflict detection.

v. NAMEX:LOGIC interface primitives.

7.8.2 COMPUTATION OF LABEL DIMENSIONS

Before any action can be taken to place a label, it is necessary to define the dimensions of a label and write these to the label record field.

```
compute_label_dimensions(Label_rec,Font,Let_sep,
    Word_sep,Line_sep,Label_leng).
```

or

```
compute_label_dimensions(Font,Let_sep,Word_sep,
    Line_sep,Label_len).
```

are called to determine the length of a specific or current label for any given character font number, and separation distances for letters, words and lines of text.

```
compute_current_label_dimensions.
```

performs the same operation but uses parameters from the current label record in the "LABEL" file.

7.8.3 COMPUTATION OF LABEL POSITIONS

A label dimensions primitive should always be called prior to computing a new position for that label, in order to update the current label variable contents.

```
compute_label_position(Fsn,Ftype,Pos,Prox,L_config).
```

is then used to compute label positions in DB_ACCESS, these can then be saved by writing the current label status register record.

Unlike point and line labels which always have a predefined fixed number of positions (20) around or along their feature, placement of labels within areas is less well defined and cannot guarantee a fixed number of positions. Therefore to find all the positions available for an area label, the following primitive must be called:

determine_area_label_positions(No).

To find out how many area label positions have been found, without re-computing the area label positions, the following primitive is used:

find_no_of_area_label_positions(No).

7.8.4 DETERMINATION OF LABELS IN POTENTIAL OVERLAP

Prior to positioning names on a map, NAMEX needs to know for any given label which other labels it could potentially overlap.

initialise_potential_labels_in_overlap.

is called to initiate this process. If at any point in the name placement process, the size of a label is forced to change due to difficulty in placement, and either its length or width increases, then the primitive must be called again.

no_of_potential_labels_in_overlap(Label,No).

returns the number of labels which can potentially overlap with a specific label and places the serial numbers of all these labels in an array, which can then be read into a list using:

get_list(No,List).

7.8.5 LABEL CONFLICT DETECTION

label_overlap(Lab1,Pos1,Lab2,Pos2,Conflict).

enlarged_label_conflict(Lab1,Pos1,Lab2,Pos2,Conflict).

are used to find out if label Lab1 at position Pos1 and

label Lab2 at position Pos2 are in overlap or conflict. If so then a "1" is returned, if not then a "0". The purpose of enlarging a label is to detect labels which, although not physically in overlap with each other, are nevertheless visually too close to one another (conflict). The enlargement is specified by the user in LOGIC (Section 7.8.6).

current_label_overlap(Lab1,Lab2,Conflict).

current_enlarged_label_conflict(Lab1,Lab2,Conflict).

perform the same operation but use the current positions of label Lab1 and label Lab2 obtained from the "LABEL" file.

no_of_labels_in_overlap(Label,No).

is used to find out the current number of labels in overlap with any specific label.

7.8.6 NAMEX:LOGIC INTERFACE PRIMITIVES

The logic and rule-based program, LOGIC, relies upon the use of many of the NAMEX database query and name

placement primitives. The NAMEX program is almost completely independent from the LOGIC program to enable a name placement system developer to concentrate on writing the LOGIC program without being restricted to a too rigid program structure.

The only restrictions are four predicates required by NAMEX which should be defined in LOGIC. The first two predicates:

initialise_name_placement_problem.

solve_name_placement_problem.

are called from menu options two and three in NAMEX and must be defined to perform their named functions. The initialisation of the name placement problem involves the selection of which names to label, selecting label configurations and positions, and determining potential label overlaps. Solving the name placement problem usually involves computing high level label information and then using this in a strategy to solve the name placement problem. It is entirely up to the writer of the LOGIC program how these predicates work. However good PROLOG programming practices, name placement efficiency and memory storage requirements should be taken into account.

The other two predicates are used in the "valid_label" database primitive (Section 7.6.5) and the "current_..." and "enlarged_label_conflict" name placement primitives (Section 7.8.5):

name_select(Name_def_rec,Ftype).

label_separation_selection(Label1,Label2).

The "name_select" primitive can be used to perform high level label selection and is independent of the selection according to feature code using the font number from the "TEXT_PARAM" file (Section 7.3.3). The label separation primitive should be defined to give horizontal and vertical buffer or separation distances between two specified labels (Section 6.6.4) and place these into the integer array in DB_ACCESS (Section 7.2).

Examples of the definition of all four name placement primitives will be given in the next chapter.

7.9 CHAPTER SUMMARY AND DISCUSSION

This chapter has described the NAMEX name placement system, designed by the author, which should be capable of placing labels on a wide variety of maps. It consists of a LOGIC program which coordinates name placement using a set of name placement rules and a name placement strategy. It is interfaced to a low level program, which accesses a specially designed name placement database and low level name placement algorithms, through the use of primitives.

The primitives are capable of performing simple name placement tasks such as generating a position for a label or detecting whether two labels overlap. However they are also used a great deal for accessing the database. This usually takes the form of simple read or write operations which can be performed either on individual fields or on lists of fields in the data files. From the existing set of primitives, it is possible to construct high level rules (Section 7.5.6).

The database utilises a combination of name orientated vector data for accessing individual features and a rasterized image of the map which can be used to supply visual data on underlying features at different label positions. A parameterized text definition rule-base is also present in the database. This gives information on the preferred usage of different name placement positions

and configurations.

Most of the information necessary for name placement is recoverable using the combined name orientated vector and raster structure of the database, although at times some extra searching and processing using the available primitives may be required. Database compaction has concentrated on the storage of raster image data in a run-length encoded form. However a reduction in the quantity of vector data can also be achieved if the user filters the source vector data (Section 3.2) prior to the generation of the NAMEX database. The database can also be accessed from external FORTRAN programs, via DB_ACCESS, in place of PROLOG.

Although a wide variety of label configurations were allowed for in the NAMEX database structure, several of these do not have any corresponding implemented algorithms. The system remains incomplete because of the sheer programming effort involved and the fact that once the use of the main name placement techniques had been proven to work, the addition of extra routines for other name configurations was of a lower scientific priority. The configurations not implemented in the system included spaced out words and letters, curved labels, arrowed labels and split labels. Such label configurations can either be included using existing PROLOG primitives to generate new primitives or by adding new subroutines to

the DB_ACCESS FORTRAN program.

The practicability of implementing the split label and curved area label configurations has already been discussed and demonstrated (Sections 6.9 and 5.4). Spaced out words and letters pose problems with testing for overlaps, in that the rectangle surrounding each word or letter in that label must be tested individually with all potential overlap labels. In the case of large spread out labels, this could be quite time consuming, therefore the approach used in rule [2.73] of placing large spread out labels first on the map would be recommended.

If the user's source cartographic database is very fast at data retrieval and they would prefer to use this instead of the NAMEX database then the DB_GENERATE program and NAMEX database should be abandoned. In place of these a translation interface directly between the DB_ACCESS program and the source database program would have to be written.

The next chapter gives some examples which illustrate how the NAMEX system can be used to tackle name placement on different types of maps.

CHAPTER 8

RULE-BASED NAME PLACEMENT USING THE NAMEX SYSTEM

8.1 INTRODUCTION

8.1.1 CHAPTER OVERVIEW

The purpose of this chapter is to illustrate how a name placement system can be implemented through the use of high level rules in a logic program. It will also demonstrate how the NAMEX system can place names on a variety of maps using three widely different examples. By studying these the user should gain an understanding of how to use the available primitives to write a logic name placement program for a particular map type. The name placement rules used have been chosen to illustrate the versatility of the NAMEX system but are probably only a small sub-set of those actually used in placing the names manually on the maps concerned.

The important factor which allows the NAMEX system to handle a large range of different map types lies mainly in the label and position/configuration selection criteria and to a lesser extent in the placement strategy. The three map examples used have nearly identical placement strategies but different label selection and

initialisation stages. The first example will describe how the system can be used to place names on the Ordnance Survey Route Planner map, containing point, line and area features. The second example places names on an administrative area map containing city point features and county boundaries. One interesting aspect of this example will be that the label size is computed automatically for different sized counties. The final example deals with the placement of crater names on a Moon map where the craters can be treated as point or area features. The name placement rules used in the Moon map example will vary with the map scale, and include abbreviation of secondary crater names.

8.1.2 A NAME PLACEMENT STRATEGY

There are many strategies for placing names on maps, many of which were described in chapter four. The NAMEX system was designed so that different name placement strategies could be easily "plugged" into the system in the form of different LOGIC programs. The strategy that LABPOS uses involves labels being placed initially in their best positions and only moved to new positions if they conflict with other labels. However, occasionally labels which are not in conflict may block others, which are in conflict, from moving to suitable positions. In all of the three logic programming examples to follow, a new

strategy will be used whereby labels will be treated as if they are going to be tested in all possible permutations of positions until a solution is found whereby no conflicts exist between labels. Due to the vast number of permutations that could result, heuristics are used to reduce the investigation of unacceptable label position permutations.

A heuristic is a means of quickening the solution to a problem, usually by reducing the amount of searching required. Four heuristics are used in the chosen strategy:

1) The first heuristic involves sorting label positions for individual names into an order of preference in a similar way to that described in section 6.5. When generating permutations of label positions, selecting highly preferred label positions first helps to find name placement solutions earlier. This is because such positions often lie in unoccupied map space or "free space" which is generally found away from high feature density areas (Rule [2.74]), and consequently less likely to conflict with other labels.

2) A second heuristic involves the way permutations are generated. A given permutation of positions is constructed one label at a time, in decreasing order of potential difficulty of placement. These labels are generally those with the least number of placement

positions, which tend to lie in feature dense areas of the map. Once a label has its position chosen, in the permutation, this restricts other labels from being placed in positions which may conflict with that label. This significantly reduces the number of positions to be investigated for the remaining labels and hence reduces the search space. The next label to be selected in the permutation will usually be a neighbour of the previous label because it will probably have had some of its positions eliminated and so will have become a difficult label to place.

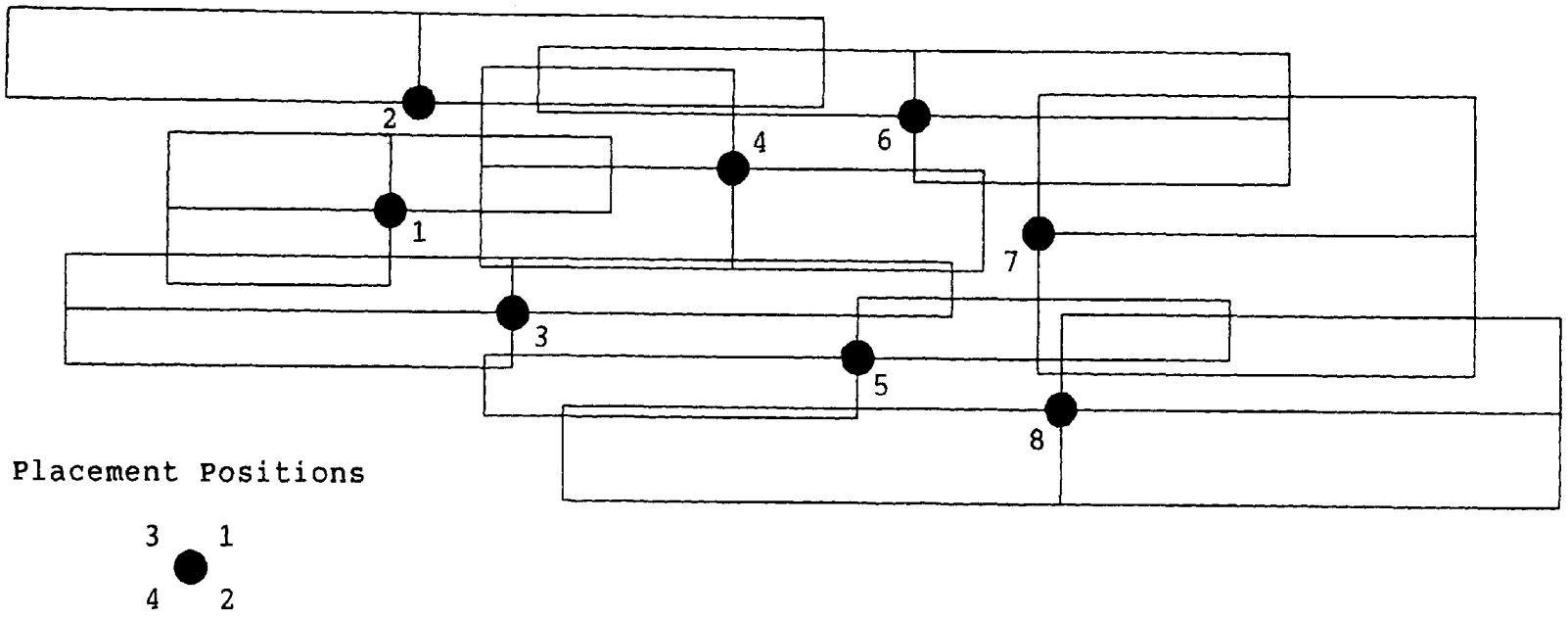
3) A third heuristic is related to the second in that during the generation of a permutation, if two or more labels have equal numbers of positions, then the label with the greatest number of potential overlaps should have its position selected first. This follows along the trend of the second heuristic in that it involves placing the most difficult labels to place first. This effectively reduces the complexity of the problem and hence the search space by limiting the possible positions of all its neighbouring labels.

4) If at any stage in the generation of a name placement permutation the heuristics fail, and result in a label with no positions to be placed in, then backtracking should occur on previously placed labels, until a stage is reached where a choice was available.

A different placement position should then be selected and the process made to continue forwards again. If labels continue to have problems with placement and no complete solution is found, this will lead back to the root label. If this occurs then the last label which was placed successfully will be treated as a "bad" label by the strategy and must be removed since it is at this point that the name placement failed. Once a label has been removed, and label overlap information re-computed, the name placement position permutation process starts from the beginning again.

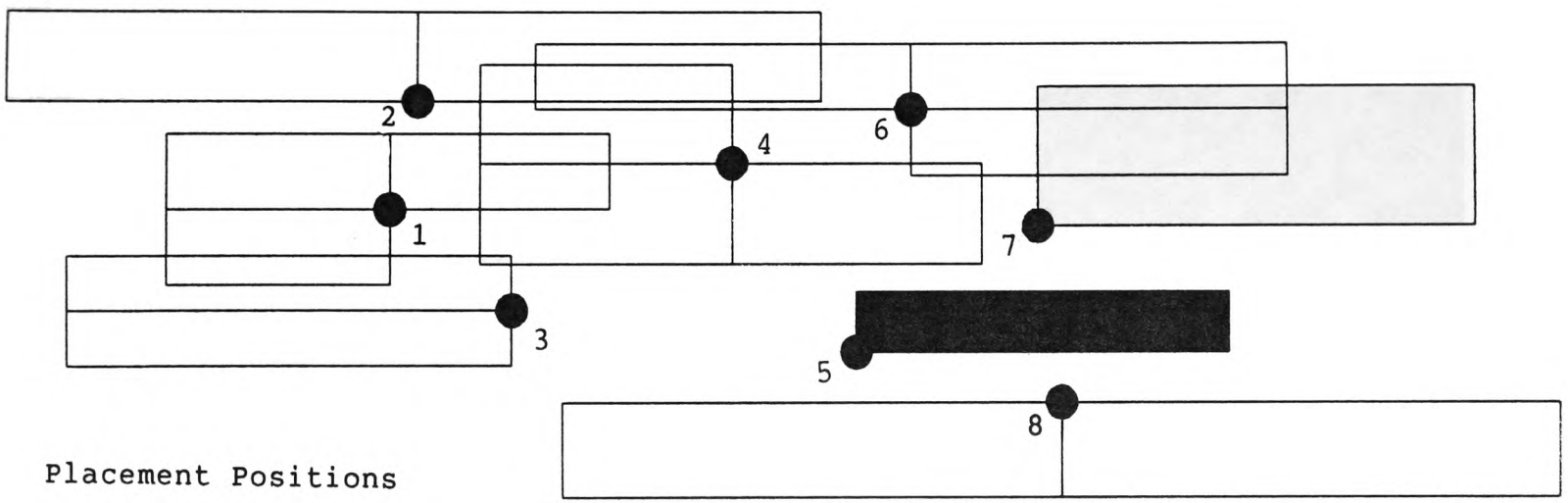
This name placement strategy is illustrated (barring backtracking and deletion) in the sequence of diagrams Fig 8.1 to Fig 8.9. For simplicity, point labels are used which are restricted to just four positions with their radius of proximity set to zero. The labels which have been placed, in the generation of this particular permutation, are shown in black and those positions still available are shown in outline.

Two data structures are present in the form of lists, one of which contains the list of positions for each label, and the other a list of potential overlapping labels. As labels are placed, both these data structures are updated by deleting information which is no longer relevant to each label.



Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3,4]	[3,4]
2	[1,3]	[4,6]
3	[1,3,4]	[1,4,5]
4	[2,3,4]	[1,2,3,6]
5	[1,4]	[3,7,8]
6	[1,2,3]	[2,4,7]
7	[1,2]	[5,6,8]
8	[1,2,4]	[5,7]

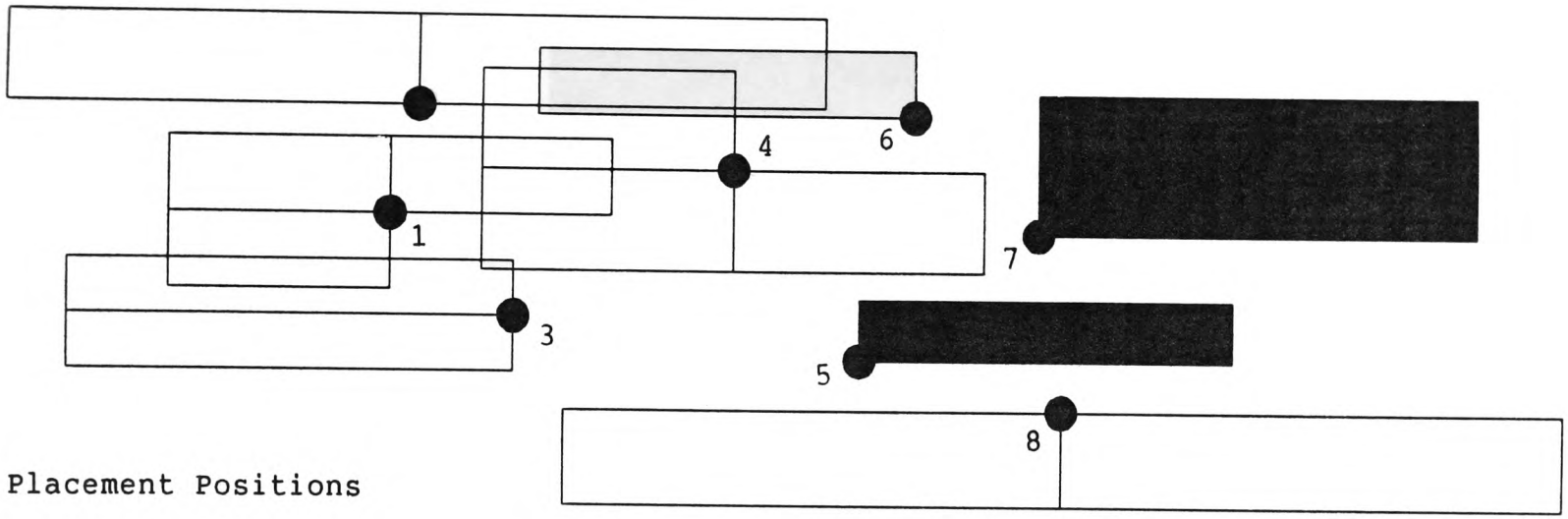
Fig 8.1 No labels placed.



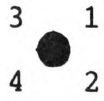
Placement Positions

Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3,4]	[3,4]
2	[1,3]	[4,6]
3	[3,4]	[1,4]
4	[2,3,4]	[1,2,3,6]
6	[1,2,3]	[2,4,7]
7	[1]	[6,8]
8	[2,4]	[7]

Fig 8.2 1 label placed.

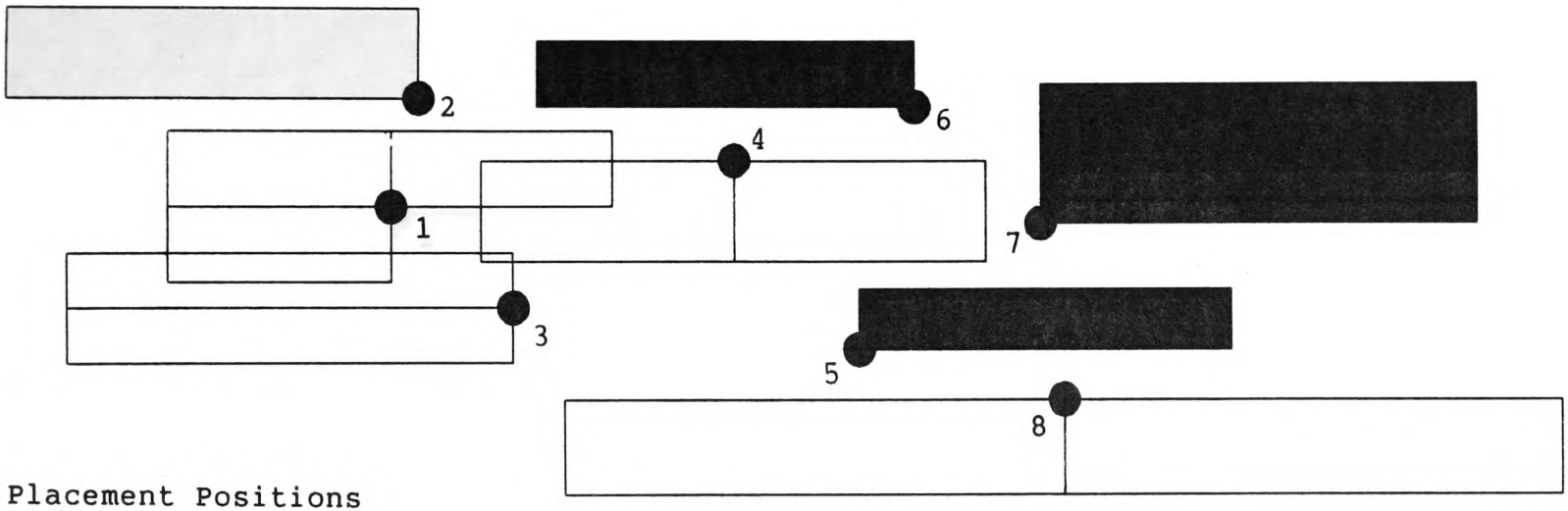


Placement Positions

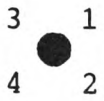


Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3,4]	[3,4]
2	[1,3]	[4,6]
3	[3,4]	[1,4]
4	[2,3,4]	[1,2,3,6]
6	[3]	[2,4]
8	[2,4]	[]

Fig 8.3 2 labels placed.

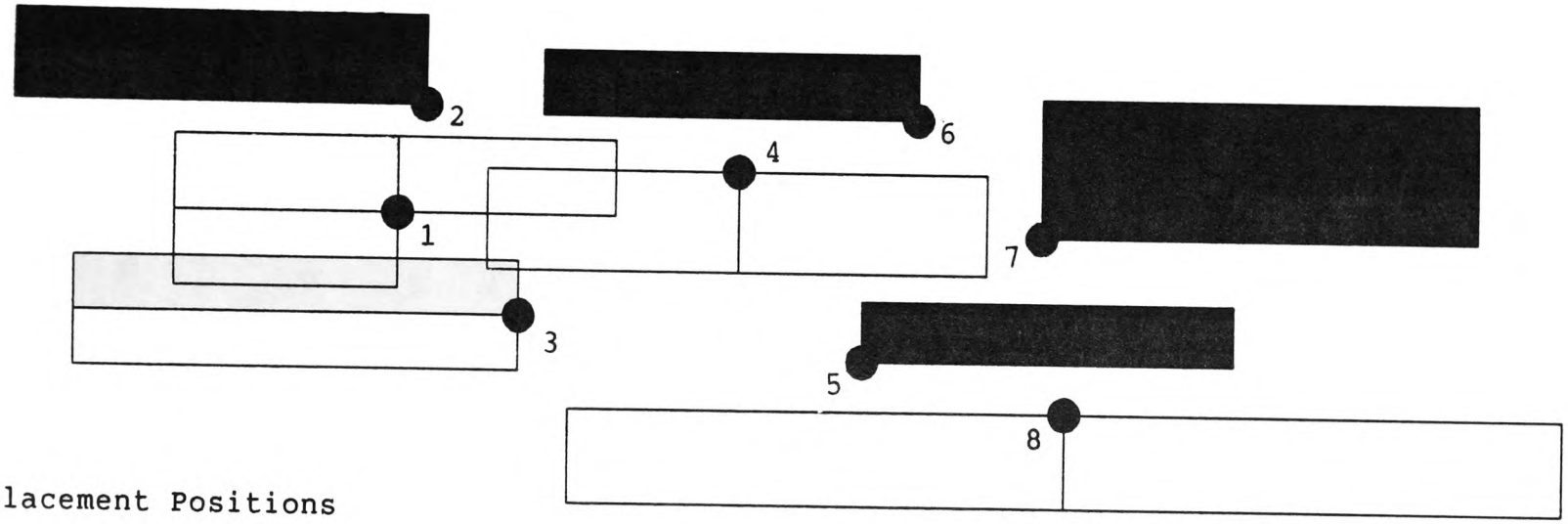


Placement Positions



Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3,4]	[3,4]
2	[3]	[4]
3	[3,4]	[1,4]
4	[2,4]	[1,2,3]
8	[2,4]	[]

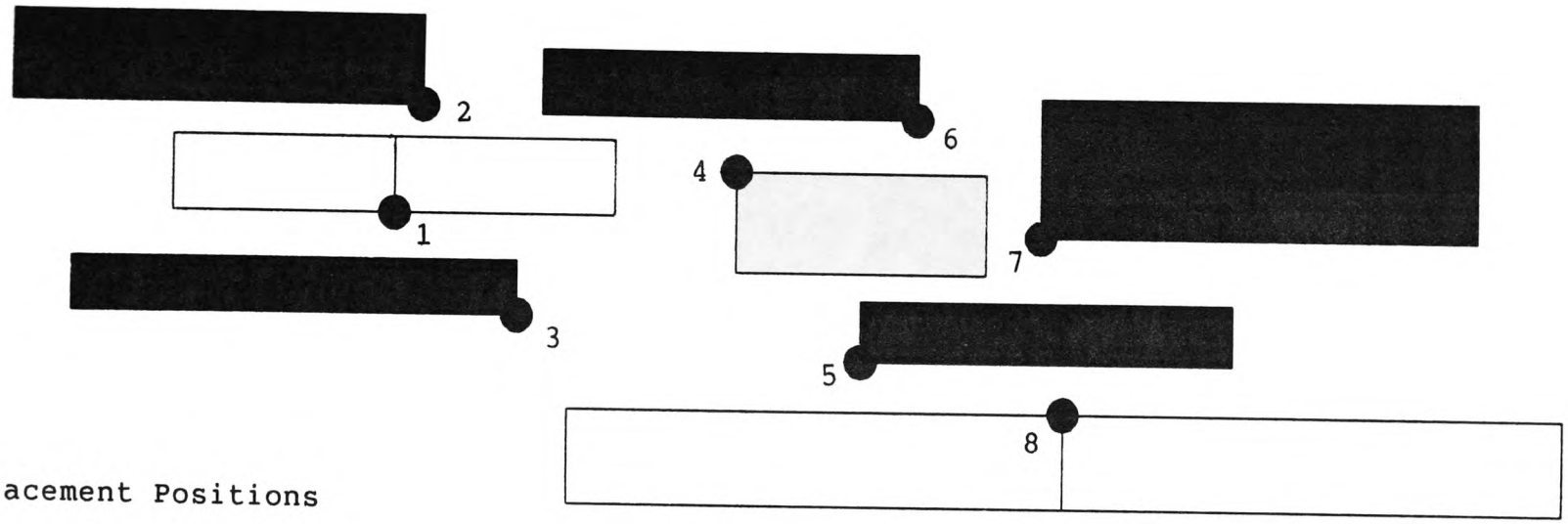
Fig 8.4 3 labels placed.



Placement Positions

Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3,4]	[3,4]
3	[3,4]	[1,4]
4	[2,4]	[1,3]
8	[2,4]	[]

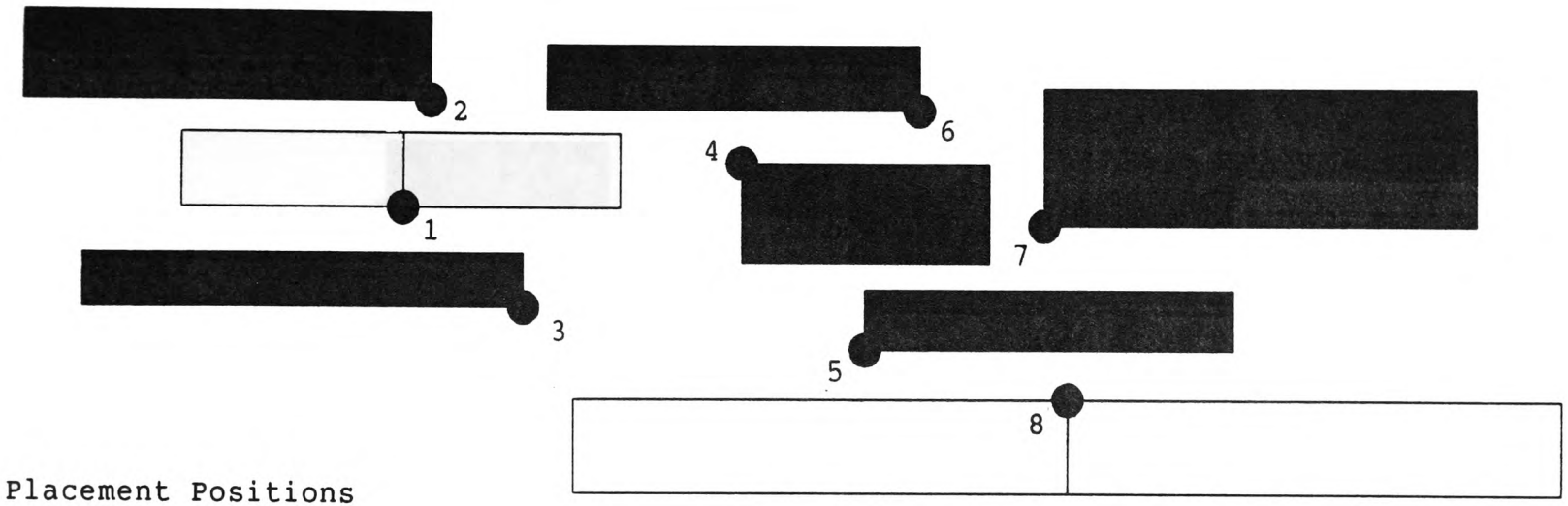
Fig 8.5 4 labels placed.



Placement Positions

Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3]	[4]
4	[2]	[1]
8	[2,4]	[]

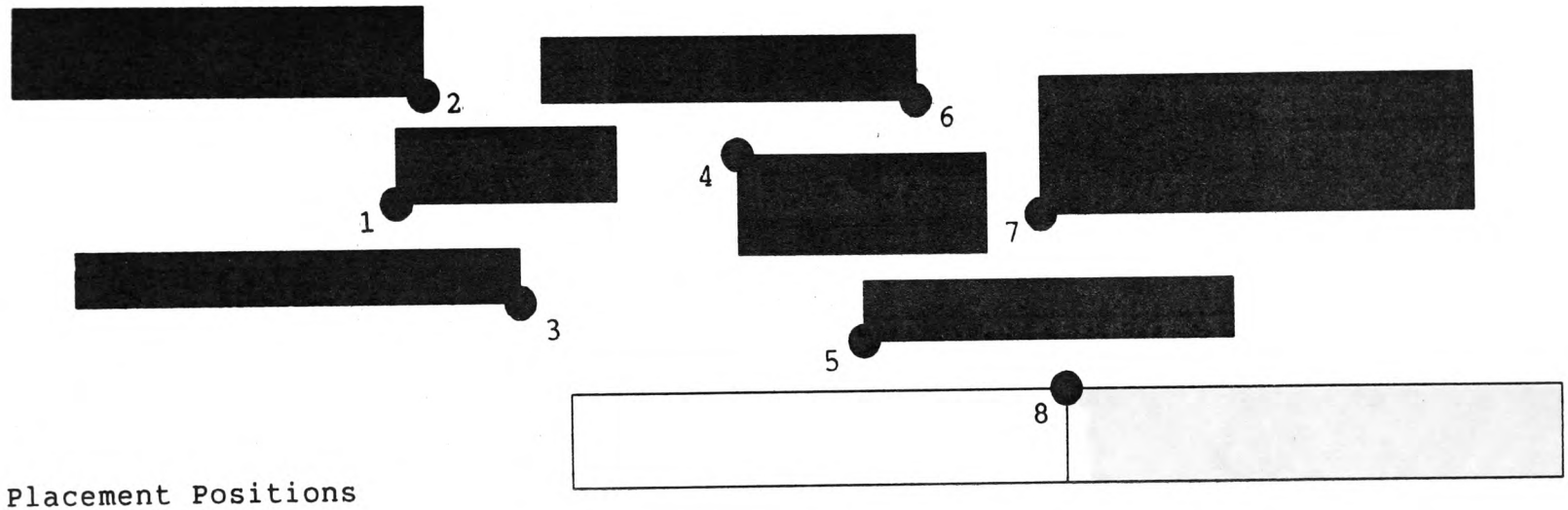
Fig 8.6 5 labels placed.



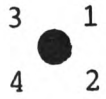
Placement Positions

Label	Order of Positions (Placement Order)	Potential Overlap Labels
1	[1,3]	[]
8	[2,4]	[]

Fig 8.7 6 labels placed.



Placement Positions



Label	Order of Positions (Placement Order)	Potential Overlap Labels
8	[2,4]	[]

Fig 8.8 7 labels placed.

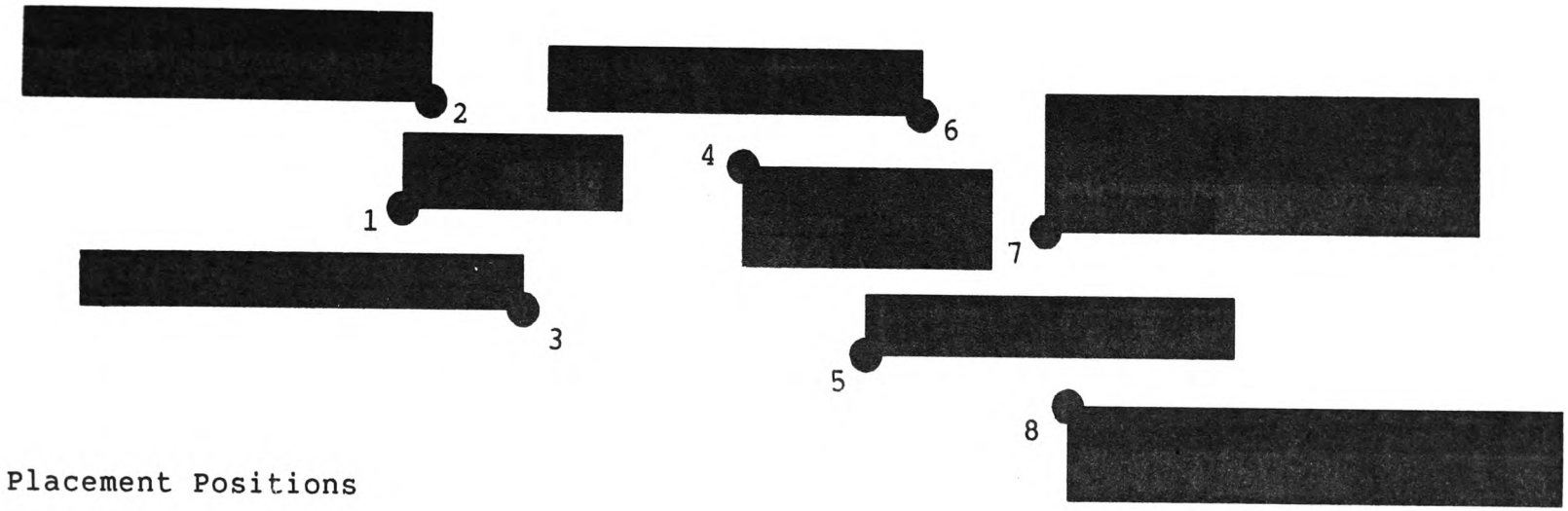


Fig 8.9 All 8 labels placed.

```
label_list( [2,-3,5,[1,4],[3,7,8]], [2,-3,7,[1,2],[5,6,8]],
  [2,-2,2,[1,3],[4,6]], [3,-4,4,[2,3,4],[1,2,3,6]],
    [3,-3,3,[1,3,4],[1,4,5]], [3,-3,6,[1,2,3],[2,4,7]],
      [3,-2,1,[1,3,4],[3,4]], [3,-2,8,[1,2,4],[5,7]] ]).
```

Fig 8.10 Label frame list for Fig 8.1.

The process starts off by considering all possible positions and potential overlaps for each label. The label with the least number of positions will be placed first, and labels in potential overlap will have their order of position and potential overlap lists updated so that the positions which remain do not overlap the placed label. The placed label has its order of position and potential overlap lists retracted once it has been successfully placed.

Fig 8.1 shows all positions available to each label. Labels 2, 5 and 7 all have the minimum number of positions, of these labels 5 and 7 have the maximum number of potential overlaps with other labels. However, label 5 has its position selected first for no other reason than it is encountered, in the table, before label 7. Therefore label 5 is placed at position 1.

In Fig 8.2, label 7 has the smallest number of label positions available and so is placed at position 1. In Fig 8.3, label 6 has only one position, and therefore must be placed at position 3. In Fig 8.4, label 2 now has only one position available, and so is also placed in position 3. In Fig 8.5, labels 3, 4 and 8 all have the same minimum number of positions, but labels 3 and 4 both have the most potential overlaps. However, because label 3 is nearest to the top of the table, it is placed in position 3. In Fig 8.6, label 4 has only one position and so must be placed

at position 2. In Fig 8.7, only two labels remain, 1 and 8, both have the same number of positions and neither have any potential overlaps. However, label 1 is selected and is placed at position 1. Finally, in Fig 8.8, the remaining label, 8, has two positions available and no overlaps. Therefore it is placed in position 2. The resulting permutation of selected label placements is shown in Fig 8.9.

For any given stage in the generation of a permutation, only a maximum of one label is removed from each potential overlap list during the updating of the label frame list. Theoretically it would be possible to reduce the potential overlap lists of labels further if those that were in potential overlap with the previously placed label, and which had their list of positions reduced, are checked to see which neighbouring labels they can still potentially overlap with. Unfortunately if this is done it is likely to increase the processing time and would make the program described in this section more difficult to understand. Therefore this has not been implemented.

8.1.3 NAMEX PREPARATION

Before NAMEX can be used to place labels on a map, preparations must be made in both software and data

domains. In particular, the LOGIC program has to be defined and this should consist of the following three parts:

i. The selection of suitable labels to place.

* ii. Generation of label details consisting of lists of preferred order of positions and potential overlap details for each label.

* iii. Label placement.

* These may involve label deletion.

An overview of the LOGIC program is illustrated in the pseudo code below:

```
BEGIN
  /* Select labels */
  for all names
    if name suitable for labelling then
      select label
    endif
  endfor

  /* Generate label details */
  for all selected labels
    generate label details
  endfor

  /* Label placement */
  dowhile label placement problem to be solved
     /* List of positions and overlap data for
        each label */
     build label frame list
     attempt to solve label placement problem
     if not(successful) then
        remove label placed last
        delete label frame list
     endif
  endwhile
  output label placement positions
END
```

The user must supply and run a data conversion program in order to convert the source cartographic database into the NAMEX data structure as described in chapter 7. A display program should also be provided so as to display the results.

Lastly several specification files must be defined using the DB_DEF program, these characterise the label placement on the map concerned:

i. "WIND_DEF" defines map window in which name placement is to take place (Section 7.5.3).

ii. "FEAT_DEF" and "RAST_DEF" define raster attributes (Section 7.5.4)

iii. "TEXT_PARAM", "TEXT_LINE" and "TEXT_AREA" define several parameterized text definition rules for labels of different feature codes (Section 7.6.3).

iv. "FONT" defines font characteristics (Section 7.6.4).

8.2 IMPLEMENTATION OF THE LOGIC PROGRAM

8.2.1 SECTION OVERVIEW

This section of the chapter describes how the LOGIC program (Section 8.1.3) is implemented in PROLOG. However, because it deals with the program at a fairly low level, section 8.2 can be skipped on a first read.

The first two parts of the implementation of the LOGIC program that will be described are label pre-processing (Section 8.2.2) and the name placement strategy in PROLOG (Section 8.2.3). These are described early on in the section whilst the high level description of the strategy is still fresh in the reader's mind. These are followed by a description of how name placement is activated (Section 8.2.4) and the selection of labels and the generation of label details (Section 8.2.5).

8.2.2 LABEL PRE-PROCESSING: CONSTRUCTION OF PRIORITY ORDERED POSITION AND POTENTIAL OVERLAP LISTS

For the implementation of the strategy, discussed in section 8.1.2, it is necessary to construct a "frame" list containing all the necessary information needed to select a permutation of label positions. The frame consists of a long list of sub-lists for each label. Each sub-list

contains the number of positions for a label, the number of labels which may potentially overlap with that label, the label number, the list of positions available and the list of potential overlaps (Fig 8.10). The sub-lists are sorted so that the most difficult labels to place are towards the head of the frame list. The length of the overlap list is made negative so that when the frame is sorted, if two labels have an identical number of positions, the one with the most potential overlaps will be selected first (Section 8.1.2, 3rd heuristic).

To construct the frame, the logic program makes use of recursive list processing of the two facts, "placement_order" (Ordered list of preferred positions) and "potential_overlap" (List of labels which may potentially overlap), which are generated by LOGIC, for each label, during the generation of label details (Section 8.2.5). All "placement_order" and "potential_overlap" data are placed into the frame list, which is contained in the fact "label_list" and built using "build_label_list" (See separate program documentation).

8.2.3 STRATEGY IMPLEMENTATION IN PROLOG

An overall view of the strategy was described in section 8.1.2 with the aid of diagrams. The same

heuristics are now written in the form of PROLOG predicates. These are called from the "name_placement_problem" predicate, described below, which generates a list of sub-lists each consisting of the number of a placed label and its position.

In "name_placement_problem", "process_list" performs the main task of recursively finding a permutation of placement positions which will not overlap other labels. Three parameters are used, firstly the label frame list, secondly a temporary list of placed labels and positions and lastly the output list. An empty list forms the basis of the temporary list, which is built up as labels and positions are selected. If this succeeds then the label output list is instantiated with the contents of the temporary list.

```
name_placement_problem(Labels_and_pos_output_list):-
   label_list(List), /* Label frame */
   /* Finds permutation of placement positions
      which do not overlap other labels */
   process_list(List,[],Labels_and_pos_output_list).
```

Before "process_list" can attain the terminating condition of having no more labels to place, it must find suitable positions for the labels, by decomposing the frame list into head and tail components. The head of the

frame list is used to obtain the label to be placed, "Lab1", its list of positions, "Poslst", and its list of overlapping labels, "Ovlst". The label being placed has its possible positions selected one by one using the "select" predicate. Although "select", like "member", has the ability to select individual elements from a list, "select" has the additional property of not causing a fail if used inside a recursive predicate that is occasionally forced to backtrack. Each time a label is processed, a record of the current label number is asserted as a fact "d(Lab1)". If the name placement attempt totally fails then this will indicate at which label the failure occurred and enable a correction to be made.

```
process_list([],Lab_output,Lab_output):- !.

process_list([[_,_,Lab1,Poslst,Ovlst]|Frm],List,Out):-
   /* Next possible position for Lab1 */
   asserta(d(Lab1)), /* Current label */
   select(Pos1,Poslst,_), /* Select a position */
   edit_n_sort_frame(Lab1,Pos1,Ovlst,Frm,New_frame),
   /* Continue with next label */
   process_list(New_frame,[[Lab1,Pos1]|List],Out).
```

The purpose of "edit_n_sort_frame" is to update the frame list each time a label is placed. It assumes that the current label can be placed until proven otherwise. By

placing a label, all labels in potential overlap are likely to have their permissible range of positions restricted. "edit_n_sort_frame" recursively decomposes the list of potential overlapping labels until either one of the labels is found to have no positions free of overlap (the predicate fails), or until there are no more potential overlap labels to test (the predicate succeeds). Each potential overlap label under consideration has its sub-list selected from the frame list so that its available position and potential overlap lists can be updated. "get_new_pos_list" performs the task of returning a new label position list containing positions which do not overlap with the current label. If the placement of the current label is valid, then the newly generated position lists will not be empty, hence the reason for the test to see if the new list length is greater than zero.

```
edit_n_sort_frame(_,_,[],Frm,Sorted_frm):-
  /* No more labels to place */
  sort(Frame,Sorted_frm),!.

edit_n_sort_frame(Lab1,Pos1,[Lab2|Tail],Frm,Sorted_frm):-
  select_first([_,_,Lab2,Poslst2,Ov_list2],Frm,New_frm),
  get_new_pos_list(Lab1,Pos1,Lab2,Poslst2,[],Newposlst2),
  length(Newposlst2,Ln_pos2),
  Ln_pos2 > 0, /* Positions still available */
  /* select_first is like select but no backtrack*/
  select_first(Lab1,Ov_list2,New_ov_list2),
  length(New_ovlp_list2,Ln_ovlp2),
  New_ln_ov2 is 0-Ln_ovlp2, /*Negate length */
  edit_n_sort_frame(Lab1,Pos1,Tail,[[Ln_pos2,New_ln_ov2,
      Lab2,Newposlst2,New_ov_list2]|New_frm],Sorted_frm).
```

If the current label is successfully placed, it is removed from its neighbours lists of potential overlap labels. The lengths of these potential overlap lists are then re-computed and negated ("New_ln_ov2") as described in section 8.2.2). If it cannot be placed then "edit_n_sort_frame" fails and "process_list" backtracks one level of recursion and tries placing the previous label in a different position (via "select").

The process of devising a new position list for labels overlapping the current label, is performed by

another recursive predicate, "get_new_pos_list", which terminates when all positions have been investigated, resulting in an empty list. On reaching this point however, the list must be reversed because the "append", used in obtaining the list, constructs it back to front. To avoid this recursive loop failing before it reaches its terminating condition, the test to find out if the current label position conflicts with Lab2 at Pos2 must not fail. If no conflict occurs, the "enlarged_label_conflict" predicate returns a "0" in a one element list, indicating that the position of the second label is valid. If they are in conflict, "1", then an empty list is returned. Whether a single element list or an empty list is returned, the "append" ensures that it is appended to the current valid list of positions.

```
get_new_pos_list(_,_,_,[],List,N_list):-
   reverse(List,N_list), !.

get_new_pos_list(Lab1,Pos1,Lab2,[Pos2|Tail],List,N_list):-
   label_test(Lab1,Pos1,Lab2,Pos2,Ans), /* Conflict ? */
   append(Ans,List,List2),
   get_new_pos_list(Lab1,Pos1,Lab2,Tail,List2,N_list).

label_test(Lab1,Pos1,Lab2,Pos2,Result):-
   enlarged_label_conflict(Lab1,Pos1,Lab2,Pos2,Ans),
   label_test2(Ans,Pos2,Result),!.
```

```
label_test2(0,Pos2,[Pos2]),!. /* No conflict */

label_test2(1,Pos2,[]),!. /* Conflict */
```

8.2.4 LABEL PLACEMENT

Name placement on a particular map is activated by calling the predicate "solve_name_placement_problem" which performs three jobs. Firstly it builds and asserts a list containing relevant label information ("label frame list"). Secondly, using this list, it attempts to solve the name placement problem and, if successful, outputs the label numbers and their selected positions into a list which is then written to the label status register (Section 7.6.2) using "fix_names_in_position". Thirdly if the attempt to solve the name placement problem is not successful, the last label to be placed (referred to as a bad label in section 8.2.5.2.3) is removed, the label frame list retracted and an empty placement order list asserted which will be removed when "solve_name_placement_problem" is run again.

```
/* Build and assert label frame list */
solve_name_placement_problem:-
   build_label_list, fail.

/* Attempt to solve name placement problem */
solve_name_placement_problem:-
   name_placement_problem(List),
   fix_names_in_position(List), !. /* See program
                                      documentation*/


solve_name_placement_problem:-
   d(Lab),  /* Last label placed (Section 8.2.3) */
   retractall(placement_order(Lab,_),/*Sect. 8.2.5.2.1*/
   /* Placement attempt failed, so remove last label
   that was placed */
   retractall(label_list(_)),
   asserta(placement_order(Lab,[])),
   remove_bad_labels, /* Section 8.2.5.2.3 */
   filter_positions_available, /* Section 8.2.5.2.4 */
   solve_name_placement_problem, !.
```

8.2.5 THE SELECTION OF LABELS AND THE GENERATION OF LABEL DETAILS

Before label placement can begin, names are selected which are suitable for labelling. Also lists of positions

available and potential overlaps with other labels must be generated for each label. Menu option 2 (Fig 7.4) is called to activate these processes through:

```
initialise_name_placement_problem:-
   select_labels, fail.

initialise_name_placement_problem:-
   generate, fail.
```

The "fail"'s are used to force both parts of the primitives to be executed and to return to the menu afterwards.

8.2.5.1 LABEL SELECTION

This involves reading through all possible names and selecting which are suitable for use as labels. During this selection process, label attributes are determined and recorded in the label status register. A pseudo logic program for label selection is given below:

```
select_labels:-
   loop_for_all_names,
      /* Based upon selected feature codes
         (Section 7.6.3) */
      validate_name_suitable_for_labelling,
      /* Label dimensions, positions,
         orientation etc */
      compute_label_attributes,
      valid_label,
      write_label_attributes,
      increment_label_record.
```

8.2.5.2 GENERATION OF LABEL DETAIL LISTS

Before the system can place names on the map, two types of fact are needed by LOGIC. These consist of the ordered list of preferred positions for each label, "placement_order" and the list of potentially overlapping labels "potential_overlaps". Additionally deletion of some labels may be required if they have no positions. Also labels with no potential overlaps with other labels are filtered to just one position, their most preferred. Generation of label details is achieved by calling:

```
generate:- generate_position_order, fail.
generate:- generate_potential_overlap, fail.
generate:- remove_bad_labels, fail.
```

```
generate:- filter_positions_available, fail.
```

8.2.5.2.1 GENERATION OF LABEL PREFERRED ORDER OF POSITION LISTS

This is implemented by calling the "generate_position_order" predicate. Initially this extracts the feature code and the number of positions available to each label (Section 7.8.3) and then masks out the current feature (Section 7.5.5). "compose_pos_list" is used to generate a list of sub-lists containing position index numbers and weights indicating placement preference. These are sorted into an order of preference and the position index numbers extracted and output as "placement_order" facts. The logic pseudo code below illustrates the generation of preferred order of position lists:

```
generate_position_order:-
   loop for all labels
      get_label_feature_code(Label,Fcode]),
      positions_available(Label,No_of_pos),
      mask_out_current_feature,
      compose_pos_list(Fcode,No_of_pos,[],List),
      sort_and_extract_weighted_pos_list(List,Poslist),
      asserta(placement_order(Label,Poslist).
```

The "compose_pos_list" predicate calls "get_priority" to extract the suggested priority of positions from the parameterized text definition rule-base. These are then used by "valid_position" to generate a list if the position is valid, constituted by a weight value indicating how preferable the position is and the position index number. If the position is invalid an empty list is returned. The returned list is then appended to the list being constructed and "compose_pos_list" recursively calls itself again until all positions have been processed at which point the complete list is returned. The logic pseudo code below illustrates how the predicate functions:

```
compose_pos_list(Fcode,0,List,List):- !.
compose_pos_list(Fcode,Pos_no,List1,List2):-
   get_priority(Fcode,Pos_no,Priority),
   valid_position(Fcode,Pos_no,Priority,Result),
   append(Result,List1,List3),
   compose_pos_list(Fcode,Pos_no-1,List3,List2).
```

The validation of label positions is discussed individually in sections 8.3.4, 8.4.4 and 8.5.4.1 since different criteria apply in each example.

8.2.5.2.2 GENERATION OF POTENTIAL LABEL OVERLAP FACTS

This is implemented by the

"generate_potential_overlap"

predicate which, after calling the

"initialise_potential_labels_in_overlap"

predicate (Section 7.8.4), loops for all labels, extracting and asserting lists of potentially overlapping labels. The pseudo logic code below illustrates this process:

```
generate_potential_overlap:-
   initialise_potential_labels_in_overlap,
   loop for all labels,
      no_of_pot_labels_in_overlap(Label,No_of_pot_ov),
      get_list(No_of_pot_ov,List),
      asserta(potential_overlap(Label,List)).
```

8.2.5.2.3 DETECTION AND REMOVAL OF BAD LABELS

Sometimes, when label position lists are generated, all the positions for a particular label are invalid due

to the presence of too much underlying detail. It is impossible to place such labels, known as "bad" labels, and so these must be removed. This is achieved by looking through all the label placement order lists and removing those with empty lists. All potential overlap lists must be checked and revised if they are found to contain the number of a label which has been removed. Pseudo logic code for removing "bad" labels is given below:

```
remove_bad_labels:-
   loop_for_all_labels,
      placement_order(Label,[]),
      potential_overlap_list(Label,Overlap_list),
      remove_placement_order(Label,_),
      remove_potential_overlap(Label,_),
      loop_for_all_potentially_overlapping_labels,
         revise_potential_overlap_list.
```

8.2.5.2.4 FILTERING OF LABEL POSITIONS

If after generating the potential overlap lists, or the removal of a label with no positions, some labels are found to have no potential overlaps with any other labels, then their most preferred position is automatically selected as their placement position and the remaining positions in their placement lists can be deleted. The effect of this is to simplify the amount of searching that

has to take place when the name placement problem is being solved. A pseudo logic program for the filtering of label positions is given below:

```
filter_positions_available:-
   for all labels,
      potential_overlap(Label,[]),
      placement_order(Label,[Best_pos|_]),
      update_placement_order(Label,[Best_pos]).
```

8.3 ROUTE PLANNER MAP

8.3.1 INTRODUCTION

The Route Planner map (Fig 2.18) has been selected in this first example because it contains a large variety of the cartographic features that are likely to be encountered by the NAMEX system and its high feature density presents a difficult challenge to the system. The point features are constituted by settlements, headlands and small islands, the line features by roads and the area features by undefined area regions which had their approximate boundaries digitised by the author from the 1:625000 scale Route Planner map.

The techniques used by the NAMEX system will differ from LABPOS in several respects in that the placement strategy (Section 8.1.2) is different, label splitting is not implemented, a small amount of area name placement occurs for unbounded geographic regions and label deletion is allowed. Also, optimization will not be investigated as thoroughly as it was with LABPOS since the general principles behind this have already been covered (Section 6.9.2).

This section of the chapter (Section 8.3) on the Route Planner map example is concerned with the map specification, the selection of labels and their

configurations, validation of label positions and the avoidance of ambiguity. These are crucial in defining the unique labelling characteristics of the map. Examples will be given of how the name placement rules can be changed and their effects on the map. A more detailed description of the LOGIC program used can be found in separate program documentation.

8.3.2 MAP SPECIFICATION

The Ordnance Survey Route Planner database is used as the primary source of data, and is supplemented with some "undefined area" data, digitised by the author. Several files have to be defined prior to converting the source cartographic data into the NAMEX format. These include the map scale and window descriptions stored in "WIND_DEF" (Table 8.1), the valid feature codes and associated descriptive names, widths and bit plane numbers held in "FEAT_DEF" (Table 8.2), and the "RAST_DEF" (Table 8.3) feature class names and bit plane priorities (See Section 7.5).

Although not required for the above data conversion process, the "TEXT_PARAM" (Section 7.6.3 and table 8.4) and "FONT" (Section 7.6.4 and table 8.5) files are needed to specify which classes of feature can be labelled and what label positions, configurations and fonts are

Table 8.1 Map window definition, "WIND_DEF", file contents for the Route Planner map grid square 0400.

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	500000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	500000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

Table 8.2 Cartographic features - contents of the "FEAT_DEF" file (excluding empty records) for the Route Planner map.

FCODE	DESCRIPTION	WIDTH (m)	BIT	FCODE	DESCRIPTION	WIDTH (m)	BIT
110	sea	100	1	332	prim_route_narrow	400	9
111	coast	100	1	335	prim_route_srv_area	1600	9
113	breakwater	100	1	337	primary_route_u_c	450	7
132	lake	0	3	340	main_road_s_c	300	11
133	reservoir	0	3	341	main_road_d_c	550	10
149	island_group	0	2	342	main_road_narrow	300	11
150	headland	0	4	347	main_road_u_c	300	7
151	geog_area	0	22	349	b_road	125	12
152	island	0	2	350	other_road	125	13
153	sands	0	2	352	motorway_tunnel	600	14
154	sea_bay	0	22	353	primary_route_tunnel	450	14
155	rocks	1500	5	354	main_road_tunnel	300	14
156	small_island	1500	5	355	b_road_tunnel	125	14
315	ferry	450	6	356	other_road_tunnel	125	14
320	motorway	600	8	361	city	5000	15
321	motorway_junction	1600	8	362	large_town	3000	16
323	mway_service_area	1600	8	363	village_on_prim_rt	1500	17
325	mway_u_c	600	7	364	town_on_prim_route	1500	18
326	mway_junction_u_c	1600	7	365	small_town_village	1500	19
327	mway_ser_area_u_c	1600	7	375	airport	1450	20
330	primary_route_s_c	450	9	376	bridge	300	21
331	primary_route_d_c	550	10				

Table 8.3 Raster classified feature groups - contents of the "RAST_DEF" file (excluding empty records) for the Route Planner map.

BIT PLANE	DESCRIPTION	PRIORITY (x10)	BIT PLANE	DESCRIPTION	PRIORITY (x10)
0	rub_out	999	15	city	999
1	sea_coast	1	16	large_town	999
2	land	0	17	prim_rt_village	999
3	lakes	1	18	prim_rt_town	999
4	headland	999	19	village	999
5	rocks	999	20	airport	999
6	ferry	10	21	bridge	10
7	under_construct	1	22	geog_area	1
8	motorway	25	23	-	-99
9	primary_route	15	24	-	-99
10	dual_carriageway	25	25	-	-99
11	main_road	10	26	-	-99
12	b_road	3	27	-	-99
13	other_road	2	28	-	-99
14	tunnel	10	29	-	-99

Table 8.4 Parameterised text definition rule-base file, "TEXT_PARAM", contents (excluding empty records and fields) for the Route Planner map.

FEATURE CODE	RADIUS PROX.	PREFERRED ORDER OF PLACEMENT POSITIONS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	FONT NO.	WORD SEP.	FEATURE TYPE
151	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	2	625	2
154	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	2	625	2
155	999	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	3	800	0
156	999	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	3	800	0
320	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	6	400	1
330	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	400	1
331	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	400	1
332	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	400	1
340	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	300	1
341	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	300	1
342	-	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	4	300	1
349	1	10	20	30	40	50	60	70	80	90	95	99	90	80	70	60	50	40	30	20	10	1	300	1
361	1144	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	7	1040	0
362	1032	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	5	938	0
363	688	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	3	625	0
364	1032	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	5	938	0
365	688	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	3	625	0
375	990	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	2	900	0
376	688	15	6	2	4	2	5	1	3	2	4	9	5	3	3	1	6	2	6	5	15	3	625	0

* KEY TO FEATURE CODES

151 = geog_area
154 = sea_bay
155 = rocks
156 = small_island
320 = motorway
330 = primary_route_s_c
331 = primary_route_d_c
332 = prim_route_narrow
340 = main_road_s_c
341 = main_road_d_c
342 = main_road_narrow
349 = b_road
361 = city
362 = large_town
363 = village_on_prim_rt
364 = town_on_prim_route
365 = small_town_village
375 = airport
376 = bridge

Table 8.5 "FONT" characteristics for the Route Planner map.

FONT NO.	BLOCK HEIGHT	BLOCK WIDTH	LETTER HEIGHT	LETTER WIDTH	LOWER CASE HEIGHT	LETTER DESCENDER HEIGHT
1	625	438	600	438	400	150
2	1100	900	900	700	500	188
3	1125	625	850	600	500	188
4	1250	800	810	625	540	202
5	1312	938	1075	688	720	269
6	1313	900	810	625	540	202
7	1375	1040	1075	750	720	269

Table 8.6 Parameterised text definition rule-base file, "TEXT_PARAM", contents (excluding empty records and fields) for the Route Planner map, reduced set of label positions.

FEATURE CODE	RADIUS PROX.	PREFERRED ORDER OF PLACEMENT POSITIONS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	FONT NO.	WORD SEP.	FEATURE TYPE
151	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	2	625	2
154	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	2	625	2
155	999	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	800	0
156	999	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	800	0
320	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	6	400	1
330	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
331	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
332	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
340	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
341	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
342	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
349	1	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	1	300	1
361	1144	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	7	1040	0
362	1032	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	5	938	0
363	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0
364	1032	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	5	938	0
365	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0
375	990	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	2	900	0
376	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0

* KEY TO FEATURE CODES

151 = geog_area
154 = sea_bay
155 = rocks
156 = small_island
320 = motorway
330 = primary_route_s_c
331 = primary_route_d_c
332 = prim_route_narrow
340 = main_road_s_c
341 = main_road_d_c
342 = main_road_narrow
349 = b_road
361 = city
362 = large_town
363 = village_on_prim_rt
364 = town_on_prim_route
365 = small_town_village
375 = airport
376 = bridge

permitted.

8.3.3 LABEL SELECTION

Although labels undergo preliminary selection on a feature code basis (Section 7.6.3), in that if no parameters are available for a particular feature code such features are not labelled, additional non-feature code selection forms a vital ingredient in defining the map content (Fig 8.11 to 8.27).

Two stages must be performed before a label can be selected, firstly the label configuration must be selected. Secondly the label is checked to see if it is suitable for placement. The process of label selection is illustrated by the logic pseudo code below:

```
select_labels:-
   loop for all names,
      select label configuration,
      validate label, /* is label OK to place ? */
      write label record.
```

8.3.3.1 LABEL CONFIGURATION SELECTION

The configurations for point, line and area labels are selected separately. Point labels only have one

configuration, horizontal, but in the case of line labels, the configuration is dependant upon the feature that is represented, the line length and its approximate angle. Area label configurations are dependant upon the principal axis and elongation of the area concerned. The chosen rules used for label configuration selection on the Route Planner map are given below [8.1] to [8.8]. A label configuration number can be selected by calling:

```
select_label_config(Fsn,Ftype,Fcode,Config).
```

The use of words instead of numbers to describe the label configurations was originally considered, however this would require an intermediate stage involving the conversion of words into corresponding integers (Table 7.14), required by the DB_ACCESS parameters, and so was not attempted.

8.3.3.1.1 POINT LABEL CONFIGURATION RULES

[8.1] The configuration of a point label is always horizontal (Configuration=0).

```
select_label_config(Fsn,0,Fcode,0):- !.
```

8.3.3.1.2 LINE LABEL CONFIGURATION RULES

In determining line label configuration, label selection also takes place in that a line feature which is shorter than a specified threshold distance (Rules [8.2] to [8.5]) will fail to have its configuration chosen and so will not be selected:

```
select_label_config(Fsn,1,Fcode,Config):-
   get_line_length(Fsn,Len),
   feat_def(Fcode,Feat_descrp,_,_), /* Section 7.5.5 */
   select_line_label_config(Feat_descrp,Fsn,Len,Config),!.
```

[8.2] A labelled "B road", which is longer than 6000m, has a diagonal configuration (Configuration=1).

```
   select_line_label_config(b_road,Fsn,Length,1):-
      Length > 6000, !.
```

[8.3] A labelled "motorway" which is longer than 7500m, has a configuration that is both horizontal and centred on the line (Configuration=7).

```
   select_line_label_config(motorway,Fsn,Length,7):-
      Length > 7500, !.
```

[8.4] A labelled "primary route single carriage way", OR a "primary route dual carriage way", OR a "narrow primary route", OR a "main road single carriage way", OR a "main road dual carriage way", OR a "narrow main road", that is longer than 6500m, AND for which the approximate angle of the road lies inside the range +/-40 degrees to the horizontal, has a configuration that is diagonal (Configuration=1).

[8.5] Same as rule [8.4] except that the approximate angle of the road lies outside the range +/-40 degrees to the horizontal. The selected configuration is both horizontal and centred on the line (Configuration=7).

Logic pseudo code for rules [8.4] and [8.5], is given below:

```
select_line_label_config(Feat,Fsn,Length,Config):-
   member(Feat,[primary_route_s_c,primary_route_d_c,
    prim_route_narrow,main_road_s_c,main_road_dc,
     main_road_narrow]),
   Length > 6500,
   compute_absolute_angle_of_line(Fsn,Angle),
   select_line_label_config(Angle,Config), !.

/* Rule [8.4], diagonal */
select_line_label_config(Angle,1):-
   Angle =< 40, !.
```

```
/* Rule [8.5], horizontal */
select_line_label_config(Angle,7):-
   Angle > 40, !.
```

8.3.3.1.3 AREA LABEL CONFIGURATION

The only area label configurations which were available to the NAMEX system, at the time of writing, were internal horizontal and diagonal configurations. These are selected as follows:

```
select_label_config(Fsn,2,Fcode,Config):-
   read_area_label_details(Fsn,[3,4],[Angle,Elong]),
   select_area_label_config(Angle,Elong,Config), !.
```

[8.6] In the case of a labelled area that has an elongation less than 0.5, the configuration is horizontal (Configuration=0).

```
select_area_label_config(Angle,Elong,0):-
   Elong < 0.5, !.
```

[8.7] In the case of a labelled area that has an elongation greater than or equal to 0.5, AND the angle of the principal axis is between +/-20 degrees to the

horizontal, the configuration is horizontal (Configuration=0).

```
select_area_label_config(Angle,Elong,0):-
    Elong >= 0.5,
    (Angle =< 20; Angle >= 340), !.
```

[8.8] Same as rule [8.7] except that the angle of the principal axis lies outside the range +/-20 degrees to the horizontal. The selected configuration is diagonal (Configuration=1).

```
select_area_label_config(Angle,Elong,1):-
    Elong >= 0.5,
    Angle > 20,
    Angle < 340, !.
```

8.3.3.2 LABEL VALIDATION

Label validation is performed with the predicate "valid_label", via "name_select(Fsn,Ftype)" (see sections 7.3.3 and 7.8.6), using specifications outlined below. All point labels are regarded as valid and similarly for line labels which have already been selected within "select_label_config". For area label positions both of the available configurations (Section 8.3.3.1.3) require

that the label fits entirely within the area, therefore at least one position must be found for an area label to be valid.

[8.9] IF the label is a point label THEN it is valid.

```
name_select(_,0):- !.
```

[8.10] IF the label is a line label THEN it is valid.

```
name_select(_,1):- !.
```

[8.11] IF the label is an area label and at least one position can be found for the label THEN it is valid.

```
name_select(_,2):-
   compute_current_label_dimensions,
   /* See section 7.8.3 */
   determine_area_label_positions(Num),
   Num > 0, !.
```

8.3.4 VALIDITY OF LABEL POSITIONS

"valid_position" (Section 8.2.5.2.1) performs a quick test to see if a label position is valid by checking that

its priority, previously extracted from the parameterized text definition rule-base, is greater than zero. If it is, then the label positional coordinates are computed and a ratio of underlying features to label area is determined. This ratio is a measure of how well placed a label is, the higher the ratio, the worse it is.

If the label represents a point feature then it must be tested to ensure that it does not encroach too close to other labelled point features. This is achieved by enlarging the rasterized rectangle under the label and performing a further test to see if at least one pixel can be found which belongs to a different labelled point feature. If so then an empty list is returned, otherwise a list containing the preference weight and position index is returned.

```
valid_position(_,_,0,[]):- !.
valid_position(Fcode,Pos,Weight,Res):-
   read_label_details(Len,Height,Prox,Ftype,Lab_area),
   compute_label_position,
   rasterize_rectangle_under_label(Len,Height,
    Underlying_feature_area),
   Ratio:= 100*Underlying_feature_area/Lab_area,
   test_larger_label(Len,Height,Prox,Ftype),
   pos_test(Ratio,Weight,Pos,Res), !.
```

```
/* For point labels allow for an enlargement
   of 1.1x radius of proximity on all sides of label */
   test_larger_label(Length,Height,Prox,0):-
      Enlg_len:= Length+2.2*Prox
      Enlg_ht:= Height+2.2*Prox
      rasterize_rectangle_under_label(Enlg_len,Enlg_ht).

/* For line and area labels */
   test_larger_label(_,_,_,Ftype):- Ftype > 0, !.
```

[8.12] IF the label represents a point feature and is too near another labelled point feature (within enlarged label rectangle) THEN that position is INVALID. The logic pseudo code for this rule is given below:

```
   /* This relies upon the pixel contents found
   with "test_larger_label" */
   pos_test(_,_,_,[]):-
      get_label_feature_type(point),
      underlying_features_one_of(city,large_town,
        prim_rt_village,prim_rt_town,village).
```

Label positions are classified as lying in "free space", "dense space" or over too much underlying detail. "Free space" can include a very small amount of underlying detail providing it is below a certain minimum percentage

threshold ratio of underlying feature area to label area. Conversely, "dense space" positions refer to second choice label positions which are acceptable providing that the percentage ratio of underlying features to the label area does not exceed the "dense_space_threshold". The two acceptable threshold ratio's are stored in the predicates:

```
free_space_threshold(5).

dense_space_threshold(75).
```

[8.13] If the label position tested lies in "free space" then the position is valid and the original parameterized text definition rule-base preference value for that position is returned. The placement order shall be as if there are no underlying features. The weight value returned is the original position priority for the label, only negated so that,when the preferred position list is sorted, all free space positions are at the front of the list. This is illustrated in the logic pseudo code below:

```
pos_test(Ratio,Priority,Pos,[Weight,Pos]):-
   free_space_threshold(Thresh_free),
   Ratio =< Thresh_free,
   Weight:= -Priority.
```

[8.14] IF after an unsuccessful application of rule [8.13] the position tested has an underlying feature to label area ratio less than or equal to the dense space thresholds, THEN the position is VALID and the underlying detail to label area ratio is returned as the weight. This is illustrated in the logic pseudo code below:

```
pos_test(Ratio,_,Pos,[Ratio,Pos]):-
    dense_space_threshold(Thresh_dense),
    Ratio =< Thresh_dense,
    free_space_threshold(Thresh_free),
    Ratio > Thresh_free.
```

[8.15] IF the position tested has an underlying feature to label area ratio which is greater than the dense space threshold THEN the position is INVALID.

```
pos_test(Ratio,_,_,[]):-
    dense_space_threshold(Thresh_dense),
    Ratio > Thresh_dense, !.
```

8.3.5 MINIMUM SEPARATION BETWEEN LABELS

When labels are tested for overlap, they are often enlarged slightly so as to allow for a minimum separation between labels. This only applies when two adjacent labels

are horizontal. The testing for overlap is performed with

"current_enlarged_label_conflict"

or

"enlarged_label_conflict"

and these require horizontal and vertical buffer or separation distances to be inserted into the first couple of elements of the integer array in DB_ACCESS (Section 7.8.6). This is achieved using "label_separation_selection" the logic pseudo code for which is given below:

```
label_separation_selection(Label1,Label2):-
   get_label_angle(Label1,Angle1),
   get_label_angle(Label2,Angle2),
   label_sep_select2(Type1,Angle1,Type2,
     Angle2,Hor_sep,Ver_sep),
   put_list([Hor_sep,Ver_sep]), !.
   /* Puts list into FORTRAN array */
```

[8.16] The minimum separation distance between two horizontal point labels should be 1.0mm vertically and 1.5mm horizontally.

```
label_sep_select2(0,0,0,0,Hor_sep,Ver_sep):-
   convert(1.0,mm,Ver_sep), /* Converts mm on the map */
   convert(1.5,mm,Hor_sep). /* into scaled metres */
```

[8.17] The minimum separation distance between horizontal point and area labels is 0.5mm vertically and 0.75mm horizontally.

```
label_sep_select2(0,0,2,0,Hor_sep,Ver_sep):-
   convert(0.5,mm,Ver_sep),
   convert(0.75,mm,Hor_sep).

label_sep_select2(2,0,0,0,Hor_sep,Ver_sep):-
   convert(0.5,mm,Ver_sep),
   convert(0.75,mm,Hor_sep).
```

[8.18] The minimum separation distance between two horizontal area labels should be as in rule [8.16].

```
label_sep_select2(2,2,Hor_sep,Ver_sep):-
   convert(1.0,mm,Ver_sep),
   convert(1.5,mm,Hor_sep).
```

8.3.6 EXAMPLES

8.3.6.1 DEFAULT SETTINGS

The first example which was attempted was of the North Devon and South Wales region using the default settings (Tables 8.1 to 8.5) and rules [8.1] to [8.18]. Unfortunately the label frame list grew so large that a stack overflow occurred and halted the program.

8.3.6.2 IMPROVING EFFICIENCY

One of the restrictions of this strategy is that there is a limit to the number of labels that can be placed. Each sub-list in the label frame list contains a list of label positions and potential overlaps, hence if there are a large number of sub-lists, the label frame list becomes very large. During backtracking in the LOGIC program, all this data is heaped onto a stack and if the number of labels typically approaches 200 in size then a stack memory overflow occurs. For this reason, the example regions from the Route Planner map to be labelled were confined to areas of low label density or where extensive label selection had been used to reduce label numbers.

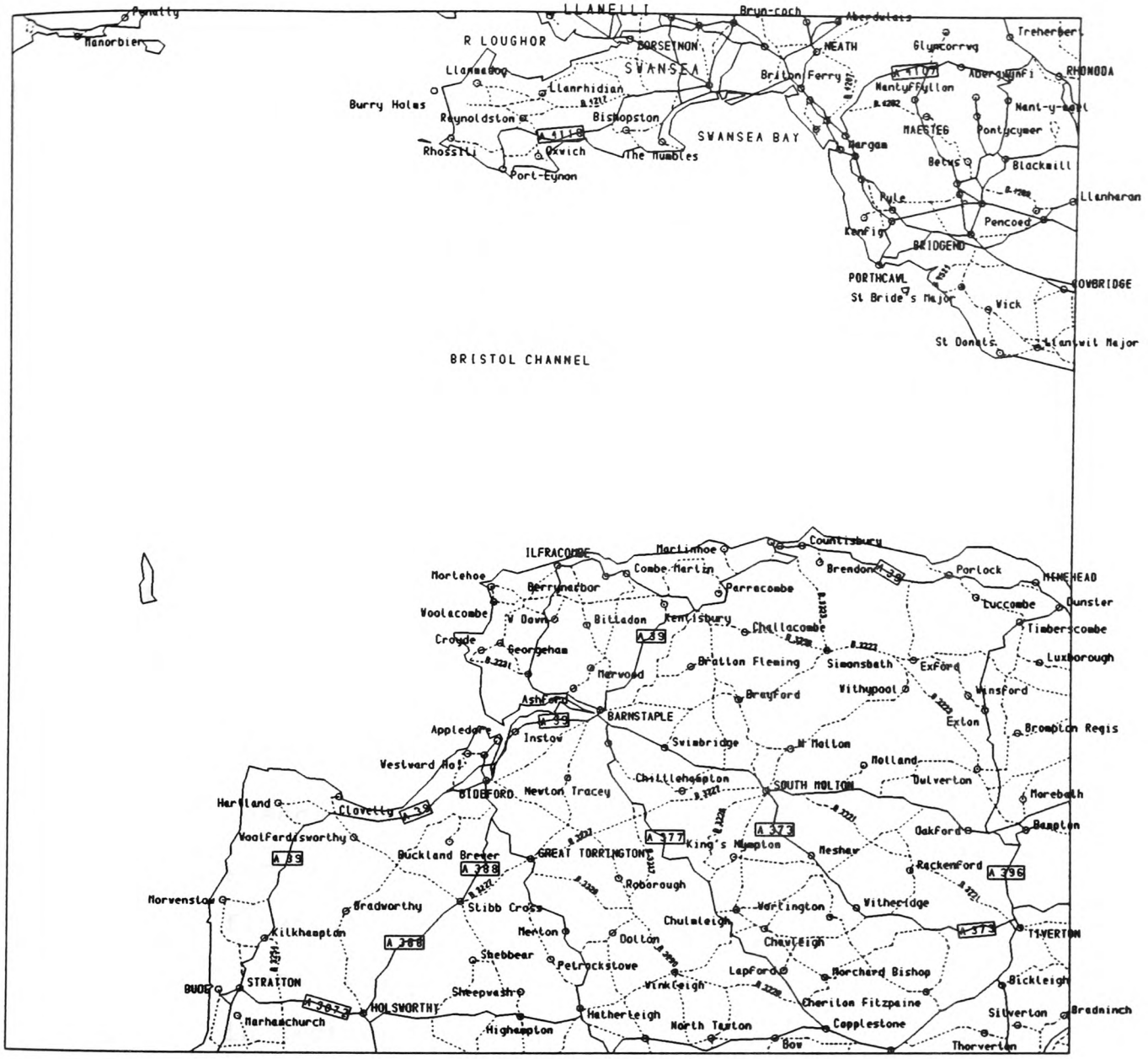
Two methods were considered for resolving this problem:

[8.19] Reducing the number of positions available to labels in the text definition rule-base to 12 say instead of 20.

[8.20] Utilizing all 20 positions in the text definition rule base, but selecting a maximum of 12 of the most preferred of these with respect to underlying detail.

Rule [8.19] was achieved by simply defining a new set of label positions (Table 8.6) and re-allocating the percentage preferences of removed positions to neighbouring positions. Fig 8.11 of Devon was successfully produced in 20 CPU minutes and yielded a name placement rate of approximately one label every 7 seconds.

Rule [8.20] was implemented by amending the LOGIC program so that the "generate_position_order" (Section 8.2.5.2) predicate only wrote a maximum of the twelve most preferred positions to the "placement_order" fact. "pull_off_list", defined in the NAMEX program, performs this function. This is illustrated in the logic pseudo code below:



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.11 Route Planner map of north Devon - default settings.

```
generate_position_order:-
   loop for all labels
      get_label_feature_code(Label,Fcode),
      positions_available(Label,No_of_pos),
      mask_out_current_feature,
      compose_pos_list(Fcode,No_of_pos,[],List),
      sort_and_extract_weighted_position_list(List,List2),
      pull_off_list(List2,Pos_list,12),
      asserta(placement_order(Label,Pos_list)).
```

When this modification was made, and the program executed, a considerable amount of backtracking took place which caused a stack overflow and the program to fail. Unfortunately, truncating the number of available positions to twelve interferes with the way the LOGIC program assesses the order of placement of labels. This depends upon placing labels with the least number of positions first (Section 8.1.2). Labels with less than 12 positions are placed in accordance with the second heuristic in section 8.1.2. However those with 12 or more positions are treated with equal preference and so the heuristic cannot be applied correctly. This results in the need for a large amount of backtracking to find a valid permutation of label positions.

In view of the success of rule [8.19] in Fig 8.11, it was used thoughout the remainder of section 8.3 (Fig 8.12 to Fig 8.27).

8.3.6.3 ALTERING "A CLASS" ROAD LABEL CONFIGURATIONS

To alter "A class" road label configuration "select_line_label_config" is amended.

[8.21] All "A class" road labels can be made horizontal (Fig 8.12):

```
select_line_label_config(Feat,Fsn,Length,7):-
    member(Feat,[primary_route_s_c,primary_route_d_c,
      prim_route_narrow,main_road_s_c,main_road_dc,
      main_road_narrow]),
    Length > 6500, !.
```

[8.22] All "A class" road labels can be made diagonal by changing the configuration number in rule [8.21] to "1" (Fig 8.13).

[8.23] "A class" road labels can be made horizontal in feature dense areas and diagonal elsewhere (Fig 8.14). Feature dense areas in this case refer to when a settlement feature lies within a circular region of radius 4km, centred on the approximate middle of the line.

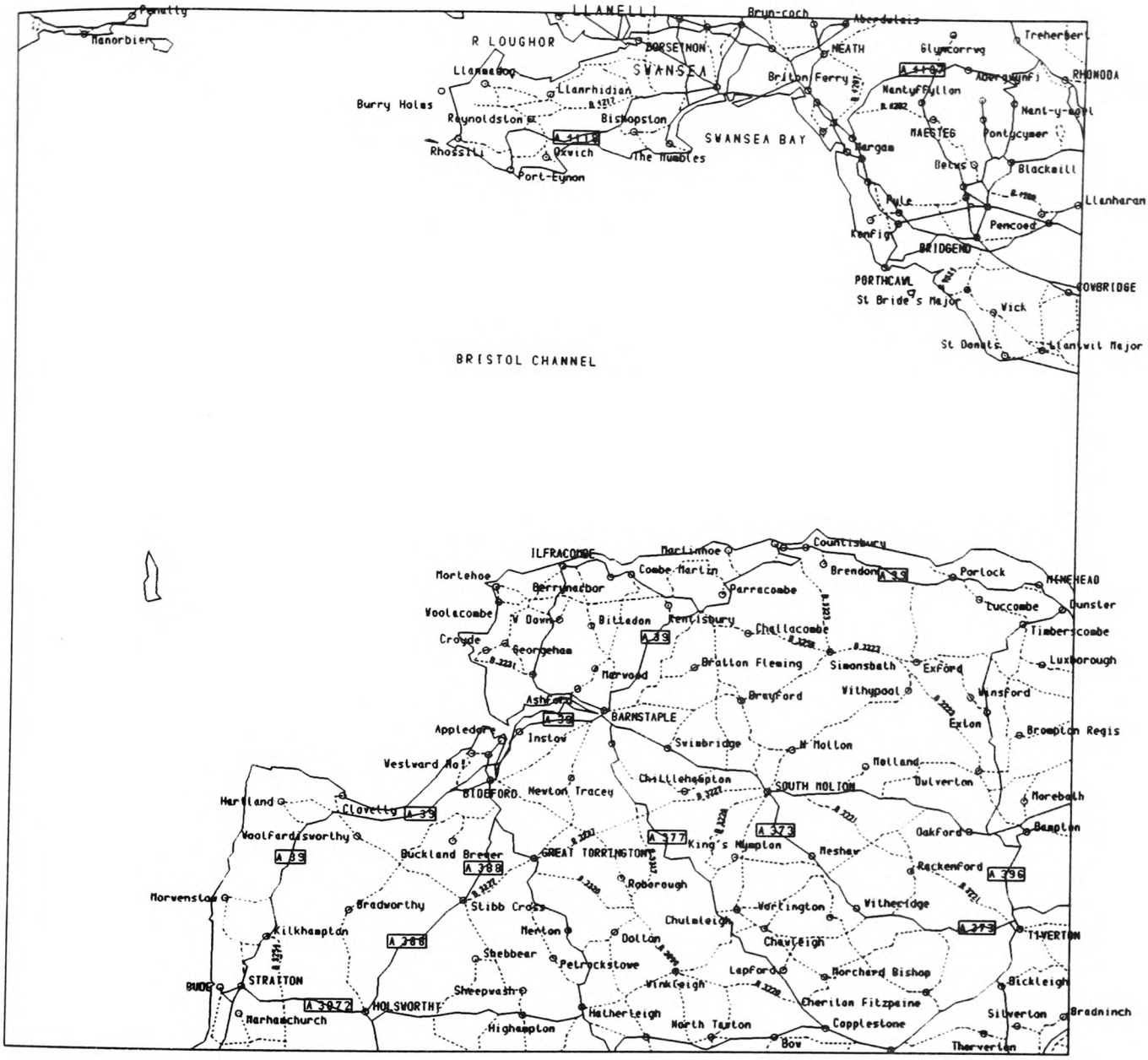
An example of the effect of this rule in Fig 8.14 is for labels in rural areas such as the "A39" near "Woolfardisworthy" and the "A39" near "Clovelly" which are

diagonally placed whereas "A class" road labels near to urban areas, for instance in South Wales, are placed horizontally.

```
select_line_label_config(Feat,Fsn,Length,Config):-
   member(Feat,[primary_route_s_c,primary_route_d_c,
     prim_route_narrow,main_road_s_c,main_road_d_c,
     main_road_narrow]),
   Length > 6500,
   get_node1(Fsn,East1,North1),
   get_node2(Fsn,East2,North2),
   East is (East1+East2)/2.0,
   North is (North1+North2)/2.0,
   /* Test within 4km (4000m) of approximate
      mid-point of line */
   rast_circle_init(East,North,4000),
   select_line_label_config(Config).

/* Horizontal if pixel discovered within 4km
   of centre of line belonging to any of these features
   (using feat_test): */
select_line_label_config(7):-
   (feat_test(city);feat_test(large_town);
     feat_test(prim_rt_village);
     feat_test(prim_rt_town);
     feat_test(village)).
```

```
/* Diagonal if no pixels discovered within 4km of
   centre of line belong to any of these features: */
select_line_label_config(1):-
   no_feat_test(city),
   no_feat_test(large_town),
   no_feat_test(prim_rt_village),
   no_feat_test(prim_rt_town),
   no_feat_test(village).

/* Pixels belonging to specified feature found */
feat_test(Feature),
   rast_def(Plane,Feature,_),
   /* Test plane to see if feature present */
   read_from_memory(Value,Plane),
   Value > 0.

/* No pixels belonging to specified feature found */
no_feat_test(Feature):-
   rast_def(Plane,Feature,_),
   read_from_memory(0,Plane).
```

Rule [8.23] was used for the remainder of section 8.3 (Fig 8.15 to Fig 8.27).



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.12 Route Planner map of north Devon - all "A class" road labels horizontal.

Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.13 Route Planner map of north Devon - all "A class" road labels diagonal.

Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.14 Route Planner map of north Devon - all "A class" road labels diagonal except in feature dense areas where they are horizontal.

8.3.6.5 INCREASING AREA LABEL SIZE [8.24]

This is achieved by changing the font characteristics for area labels (Tables 8.7 and 8.8). The new font characteristics are used throughout the remainder of section 8.3 (Fig 8.15 to Fig 8.27).

8.3.6.6 EXCLUDING B CLASS ROAD LABELS [8.25]

This is simply achieved by removing all occurrences of B class road labels from the TEXT_PARAM file and/or removing rule [8.2] (Fig 8.15).

8.3.6.7 VARIABLE POINT RADIUS OF PROXIMITY [8.26]

Settlement labels in crowded areas of the map (where a labelled settlement is within 4km of another settlement) have their radius of proximity halved to ease placement (Fig 8.16 to 8.18 and subsequent maps in section 8.3). The predicate "determine_prox(Ftype,Fsn,Prox,New_prox)" is used to achieve this and is called from the label selection predicate during the computing of label attributes (Section 8.2.5.1 and separate program documentation).

Table 8.7 Parameterised text definition rule-base file, "TEXT_PARAM", contents (excluding empty records and fields) for the Route Planner map, - increased area label font size (8).

FEATURE CODE	RADIUS PROX.	PREFERRED ORDER OF PLACEMENT POSITIONS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	FONT NO.	WORD SEP.	FEATURE TYPE
151	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	8	625	2
154	-	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	8	625	2
155	999	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	800	0
156	999	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	800	0
320	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	6	400	1
330	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
331	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
332	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	400	1
340	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
341	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
342	-	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	4	300	1
349	1	0	30	0	0	0	40	0	0	0	50	0	0	0	40	0	0	0	30	0	0	1	300	1
361	1144	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	7	1040	0
362	1032	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	5	938	0
363	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0
364	1032	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	5	938	0
365	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0
375	990	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	2	900	0
376	688	15	6	0	8	0	5	0	6	0	0	13	5	0	7	0	7	0	7	5	15	3	625	0

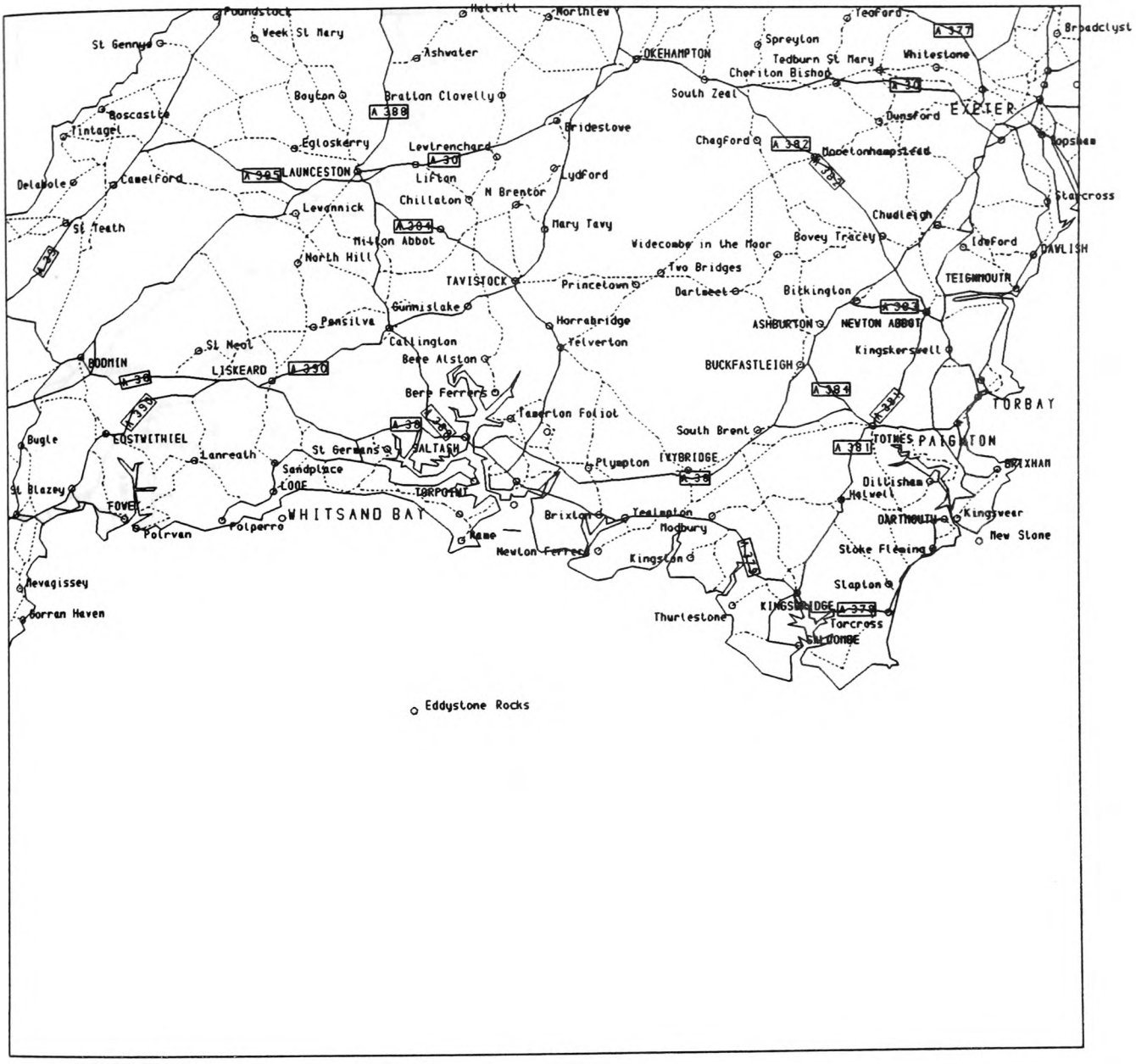
* KEY TO FEATURE CODES

151 = geog_area
154 = sea_bay
155 = rocks
156 = small_island
320 = motorway
330 = primary_route_s_c
331 = primary_route_d_c
332 = prim_route_narrow
340 = main_road_s_c
341 = main_road_d_c
342 = main_road_narrow
349 = b_road
361 = city
362 = large_town
363 = village_on_prim_rt
364 = town_on_prim_route
365 = small_town_village
375 = airport
376 = bridge

Table 8.8 "FONT" characteristics for the Route Planner map - increased font (8) size for area labels

FONT NO.	BLOCK HEIGHT	BLOCK WIDTH	LETTER HEIGHT	LETTER WIDTH	LOWER CASE HEIGHT	LETTER DESCENDER HEIGHT
1	625	438	600	438	400	150
2	1100	900	900	700	500	188
3	1125	625	850	600	500	188
4	1250	800	810	625	540	202
5	1312	938	1075	688	720	269
6	1313	900	810	625	540	202
7	1375	1040	1075	750	720	269
8	1375	1125	1125	875	625	235

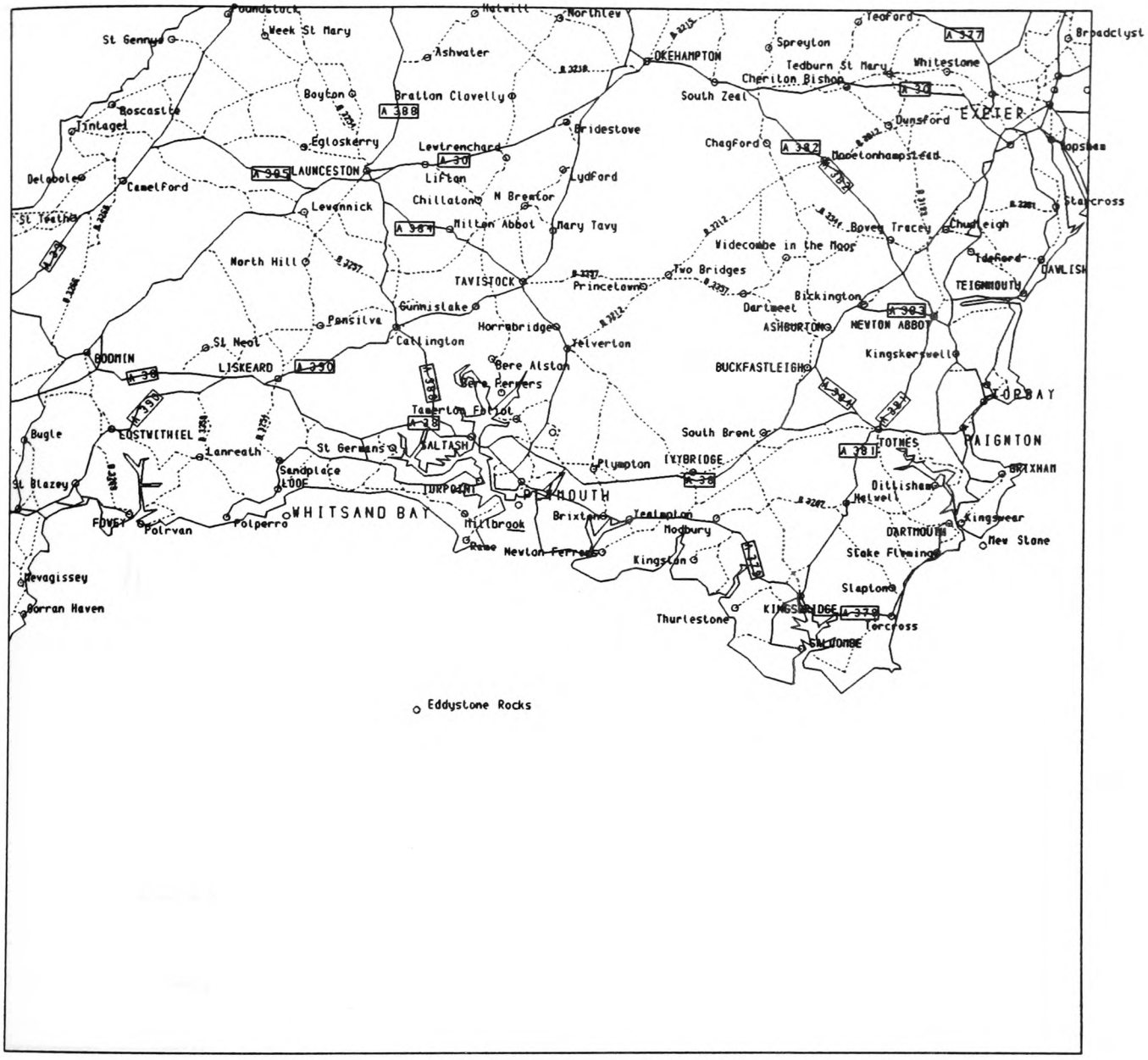
```
determine_prox(Fsn,0,Prox,New_prox):-
   get_point_coords(Fsn,East,North),
   mask_out_current_feature,
   rast_circle_init(East,North,4000),
   determine_point_prox(Prox,New_prox).

/* For line and area labels the radius of
   proximity is not used so the original
   value is returned */
determine_prox(_,1,Prox,Prox).
determine_prox(_,2,Prox,Prox).

/* Radius of proximity stays the same if settlement
   does not lie in a crowded area of the map */
determine_point_prox(Prox,Prox):-
   no_feat_test(city),
   no_feat_test(large_town),
   no_feat_test(prim_rt_town),
   no_feat_test(prim_rt_village).

/* Radius of proximity halved if settlement
   lies in a crowded area of the map */
determine_point_prox(Prox,Half_prox):-
   (feat_test(city); feat_test(large_town);
     feat_test(prim_rt_town); feat_test(prim_rt_village);
     feat_test(village)), /* Section 8.3.6.4 */
   Half_prox is Prox/2.
```



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	0	m
JEAST	Map Window	300000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

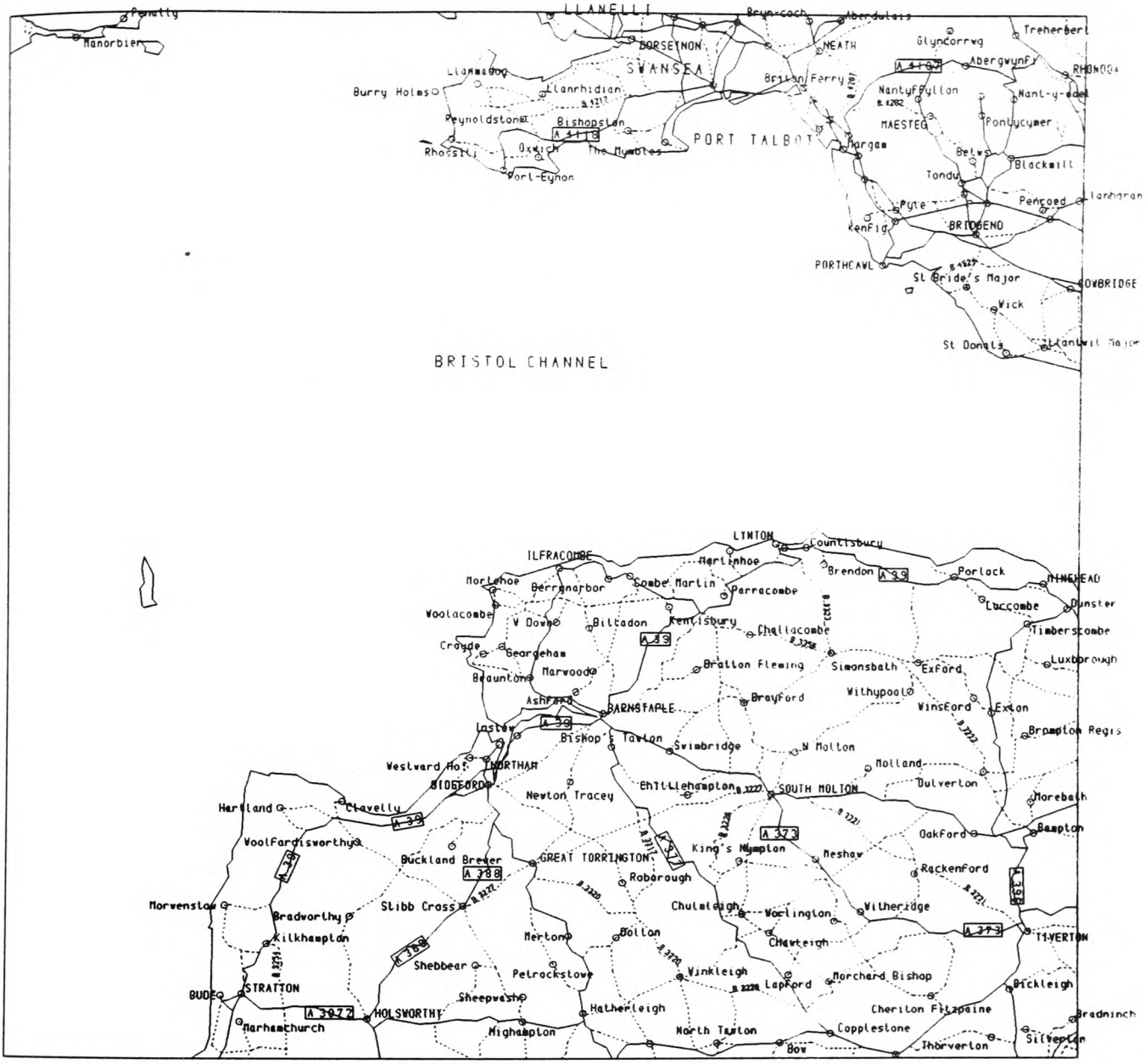
Fig 8.15 Route Planner map of south Devon - all "B class" road labels removed.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	0	m
JEAST	Map Window	300000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

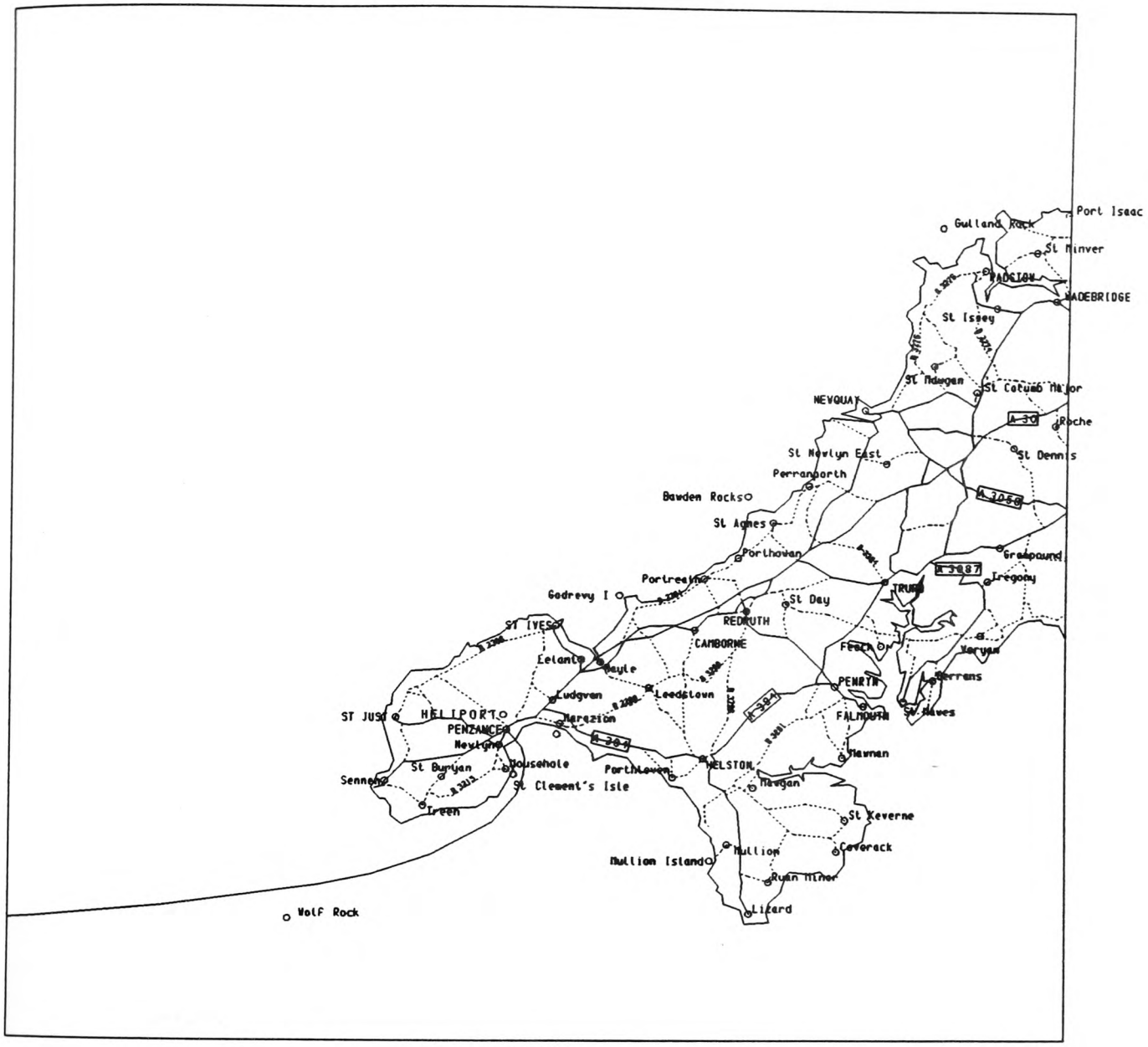
Fig 8.16 Route Planner map of south Devon - variable point label radius of proximity. 464



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.17 Route Planner map of north Devon - variable point label radius of proximity.



Window Definition File "WIND_DEF" contents

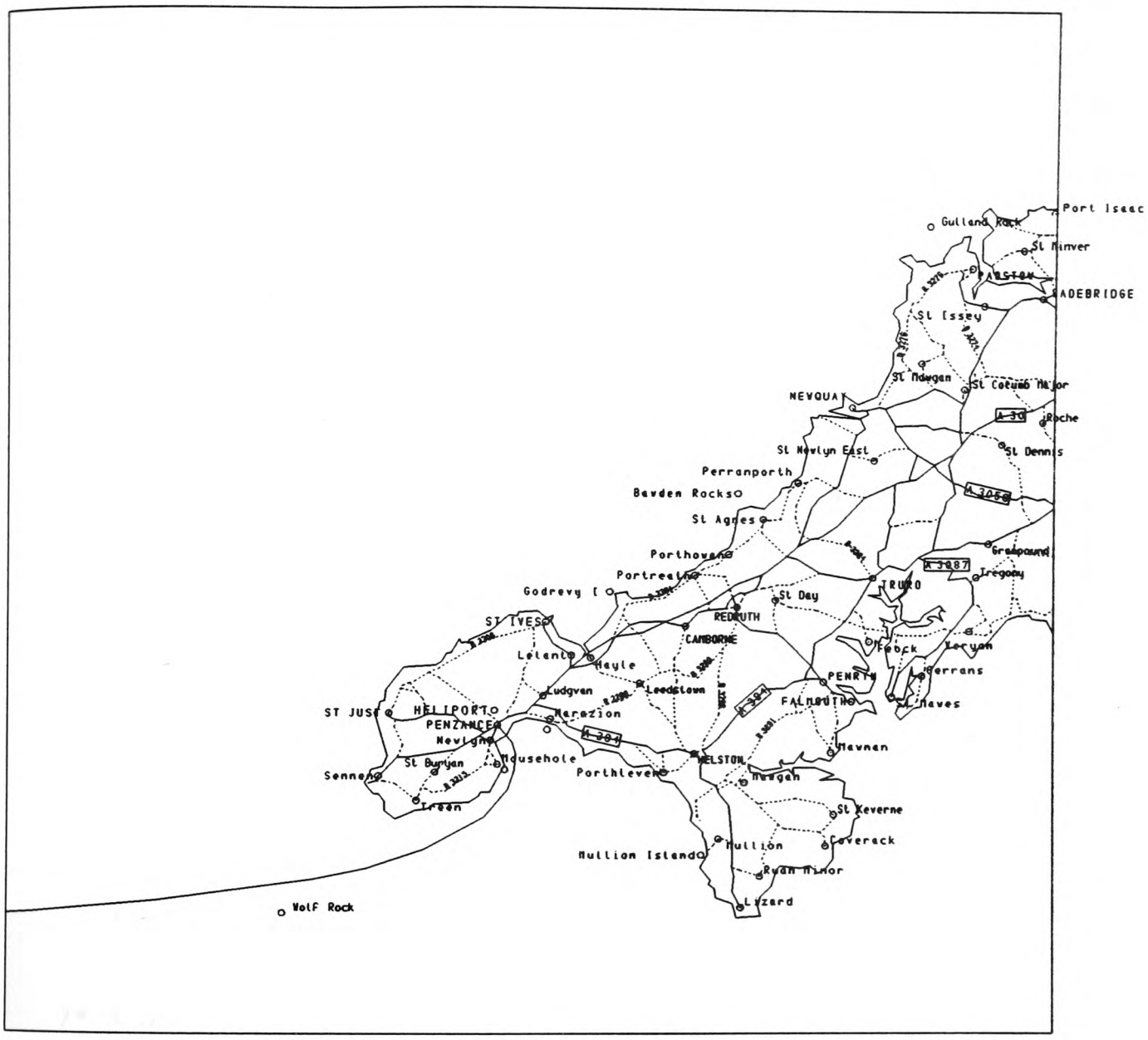
RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	100000	m
INORTH	Map Window	0	m
JEAST	Map Window	200000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	100000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	200000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

Fig 8.18 Route Planner map of Cornwall - variable point label radius of proximity.

8.3.6.8 INCREASE THE SIZE OF COASTAL SETTLEMENT LABELS [8.27]

This is achieved by testing for coastline within say a radius of 2km of a settlement using rule [2.22] encoded as the prolog predicate "rule2_22(Ftype,Fsn,Font, New_font)" (Section 7.5.6). If a settlement is found to lie near the coastline then providing that font characteristics are stored in the order of increasing character size, an enlargement of the label can simply be achieved by increasing the recommended font number by one (Fig 8.19). This rule is used in the remaining examples in section 8.3 (Fig 8.20 to Fig 8.27):

```
rule2_22(0,Fsn,Font,New_font):-
   get_point_coords(Fsn,East,North),
   rast_circle_init(East,North,2000),
   rast_def(Plane,sea_coast,_),
   read_from_memory(Value,Plane),
   rule2_22a(Value,Font,New_font).

/* Settlement not within 2km of coast */
rule2_22a(0,Font,Font).
/* Settlement within 2km of coast */
rule2_22a(Value,Font,New_font):-
   Value>0,
   New_font is Font + 1.
```



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	100000	m
INORTH	Map Window	0	m
JEAST	Map Window	200000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	100000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	200000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

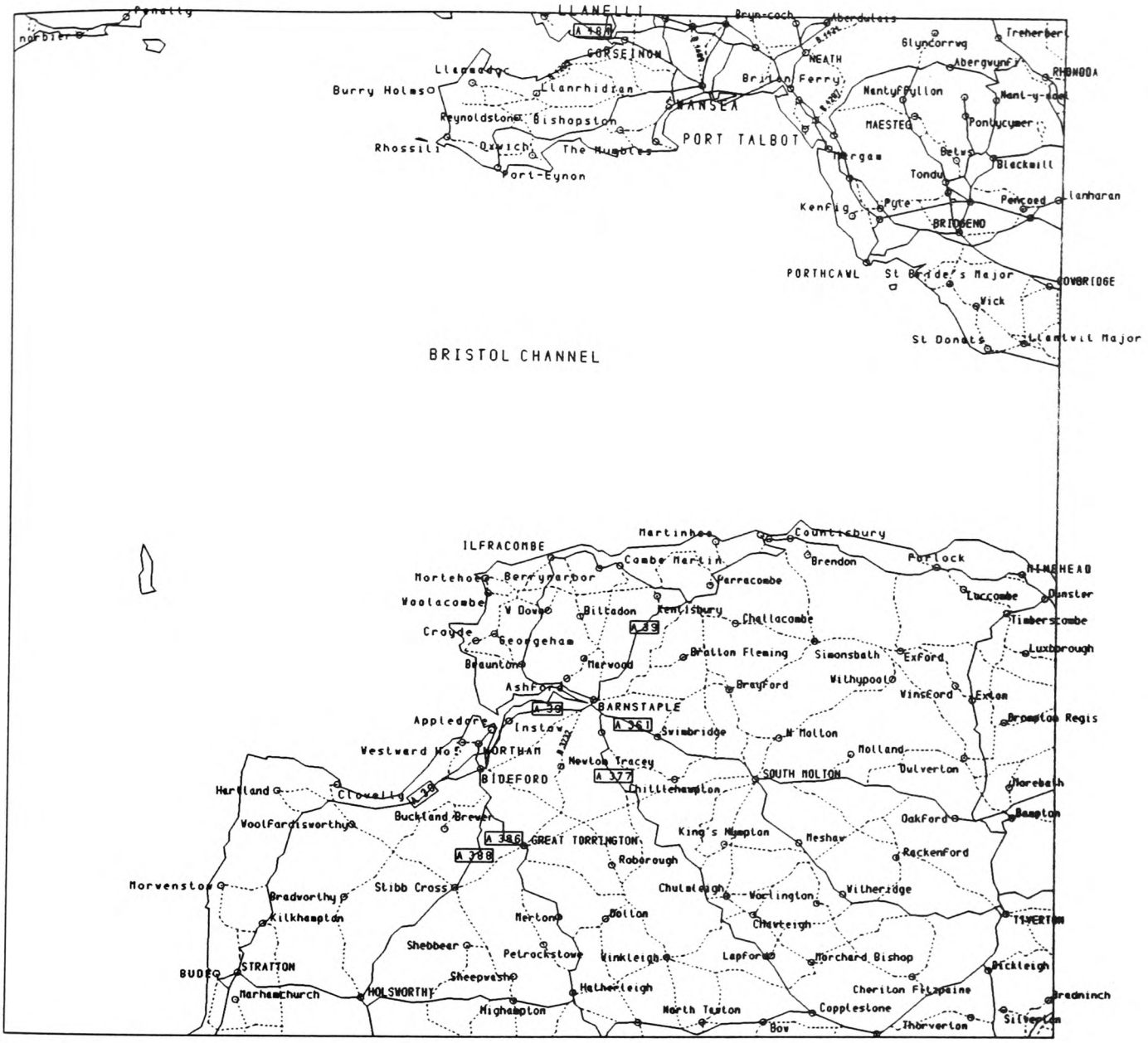
Fig 8.19 Route Planner map of Cornwall - settlement near coast have increased label size.

```
   /* Don't increase font size if line or area feature */
   rule2_22(1,_,Font,Font).
   rule2_22(2,_,Font,Font).
```

8.3.6.9 ONLY LABEL ROADS AT IMPORTANT JUNCTIONS [8.28]

This can be achieved by testing to see if either end of a road, is adjacent (within 5km) to a settlement other than a village (Fig 8.20 and 8.21). If so it is selected for labelling. A modification to "select_label_config" (Section 8.3.3.1.2) is made in that "check_end_nodes(Fsn)" is added on after "select_line_label_config":

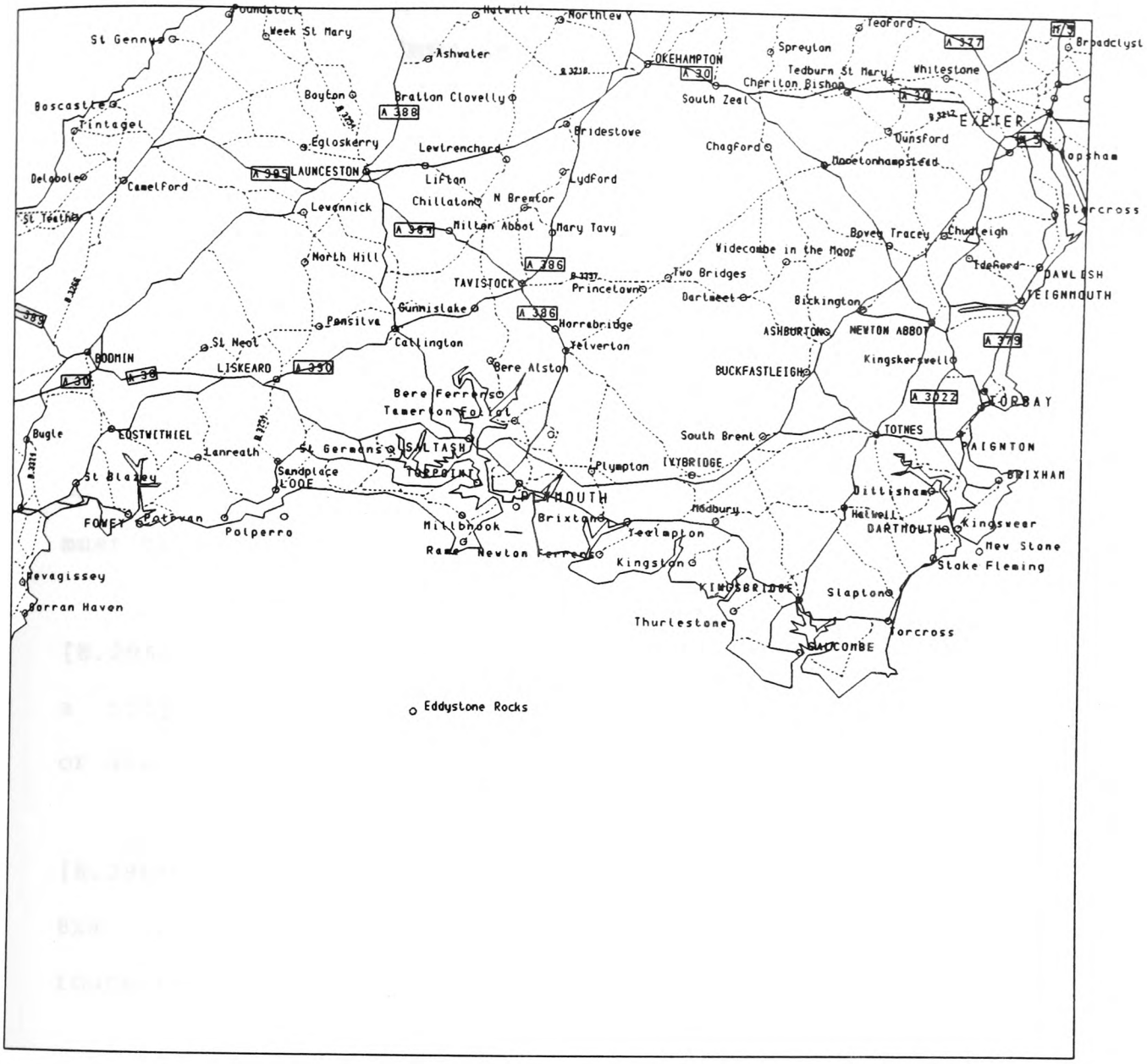
```
check_end_nodes(Fsn):-
   get_node(Fsn,East1,North1),
   get_node(Fsn,East2,North2),
   (important_node(node1); important_node(Node2)).

/* A node is important if it lies within 5km of
   a settlement other than a village */
important_node(Fsn):-
   get_point_coords(Fsn,East,North),
   rast_circle_init(East,North,5000),
   (feat_test(city); feat_test(large_town);
   feat_test(prim_rt_town); feat_test(prim_rt_village)).
```



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.20 Route Planner map of north Devon - only label roads at important junctions.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	0	m
JEAST	Map Window	300000	m
JNORTH	Map Window	100000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	100000	m
SCALE	Map Scale	625000	

Fig 8.21 Route Planner map of south Devon - only label roads at important junctions.

8.3.6.10 DELETE LABELS IF TOO NEAR IMPORTANT SETTLEMENT FEATURES

Although the source cartographic database from which the NAMEX database was derived (the Ordnance Survey 1:625000 Route Planner database in this case) is already generalised for the map scale concerned, it may sometimes be necessary to filter labels. This is necessary when applying the current placement strategy to label dense areas such as London or Bristol, where a stack overflow must be avoided.

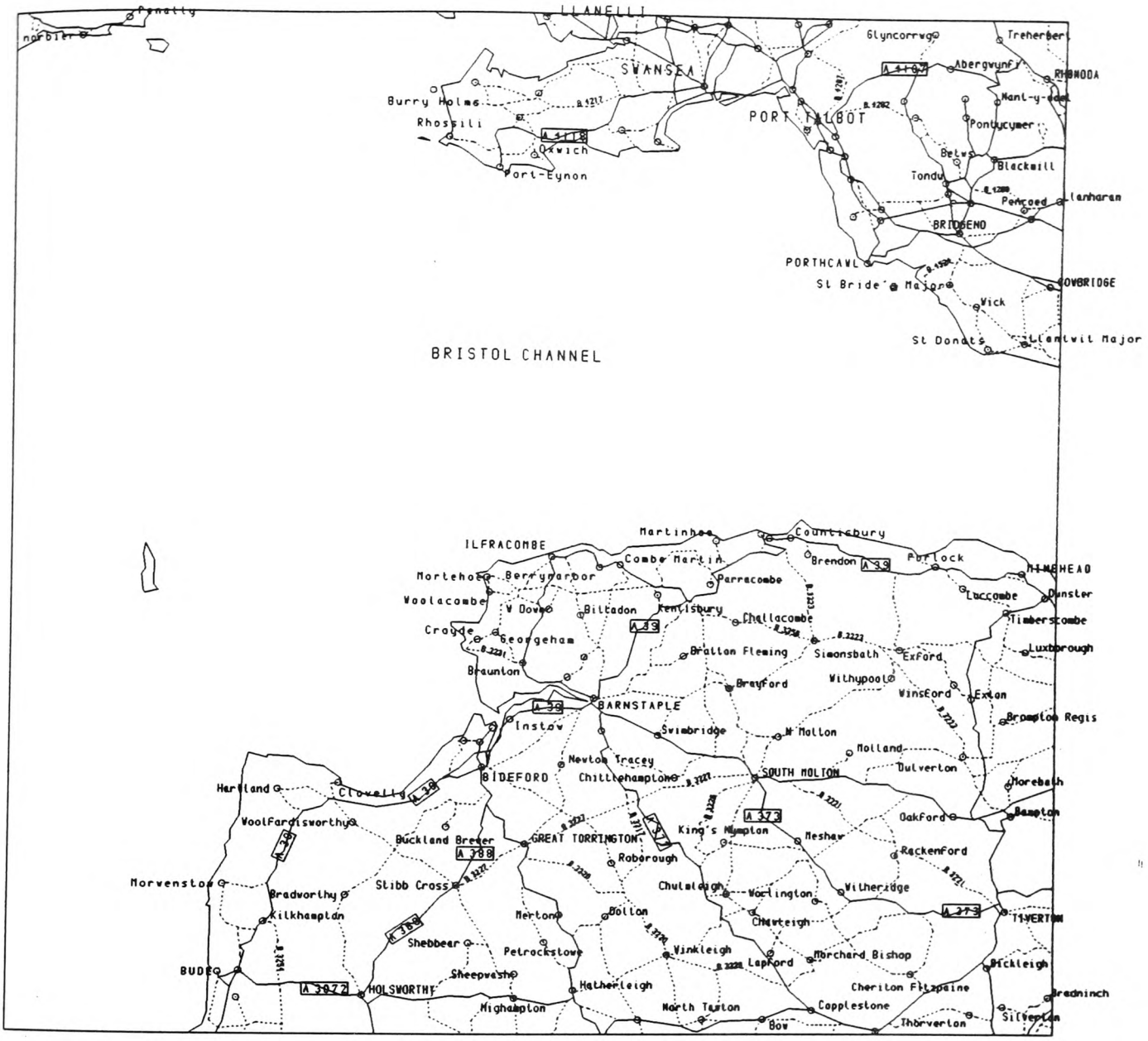
[8.29A] No small town/village to be placed within 10km of a city, 8km of a large town, 6km of a primary route town or 4km of a primary route village.

[8.29B] No village on a primary route to be placed within 8km of a city, 6km of a large town or 4km of a primary route town.

[8.29C] No town on a primary route to be placed within 6km of a city or 4km of a large town.

[8.29D] No large town to be placed within 4km of a city.

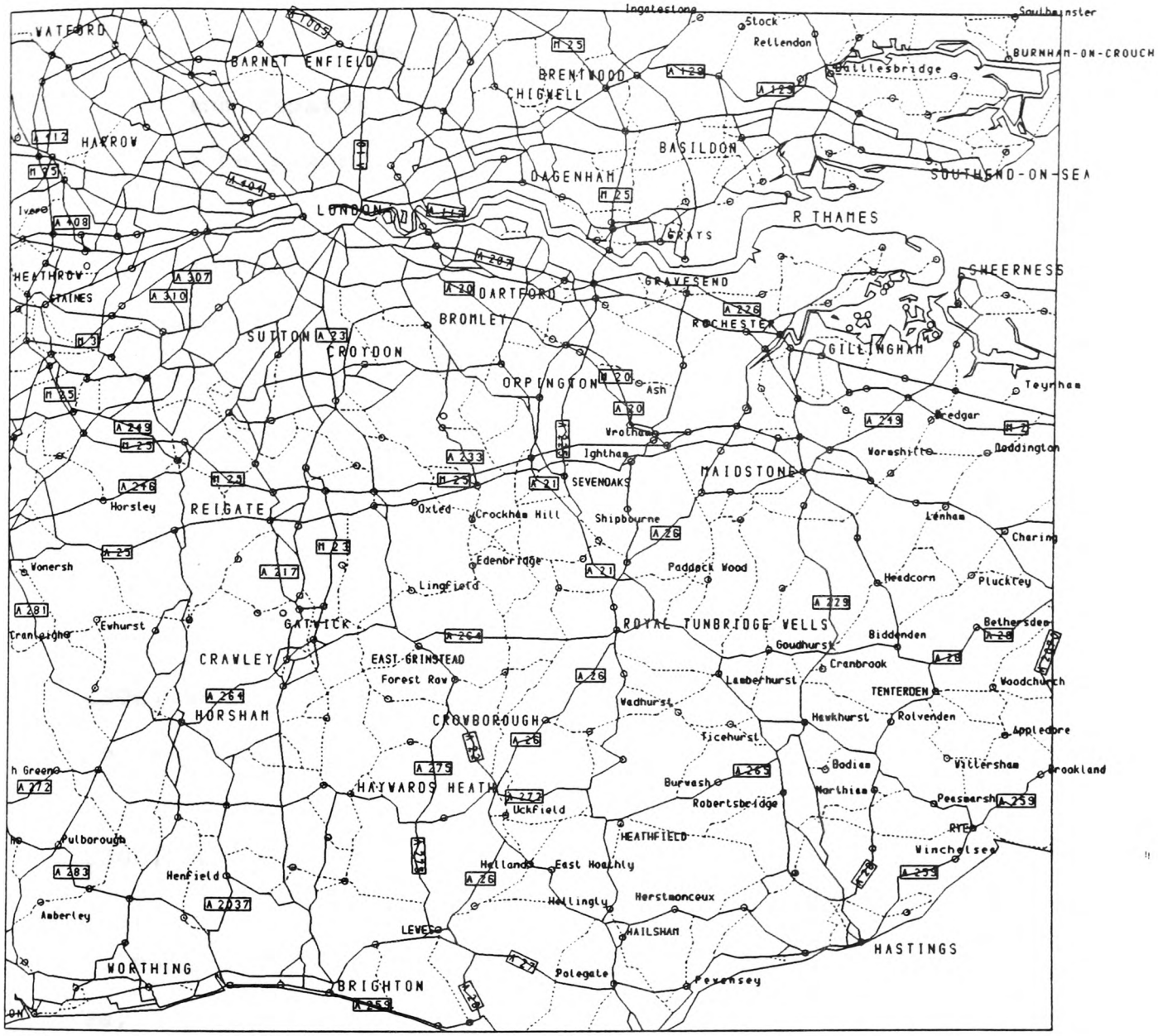
Rules [8.29A] to [8.29D] were applied to Fig 8.22, and Fig 8.23. The latter had all B road labels removed to reduce label numbers.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	200000	m
INORTH	Map Window	100000	m
JEAST	Map Window	300000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	200000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	300000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.22 Route Planner map of north Devon - do not label villages within 8km of large towns etc.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	500000	m
INORTH	Map Window	100000	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	500000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.23 Route Planner map of London - do not label villages within 8km of large towns etc.

In Fig 8.24, the rules were re-defined by reducing the minimum separation distances so as to allow more rural labels to be placed:

[8.30A] No small town/village to be placed within 7km of a city, 6km of a large town, 5km of a primary route town or 4km of a primary route village.

[8.30B] No village on a primary route to be placed within 6km of a city, 5km of a large town or 4km of a primary route town.

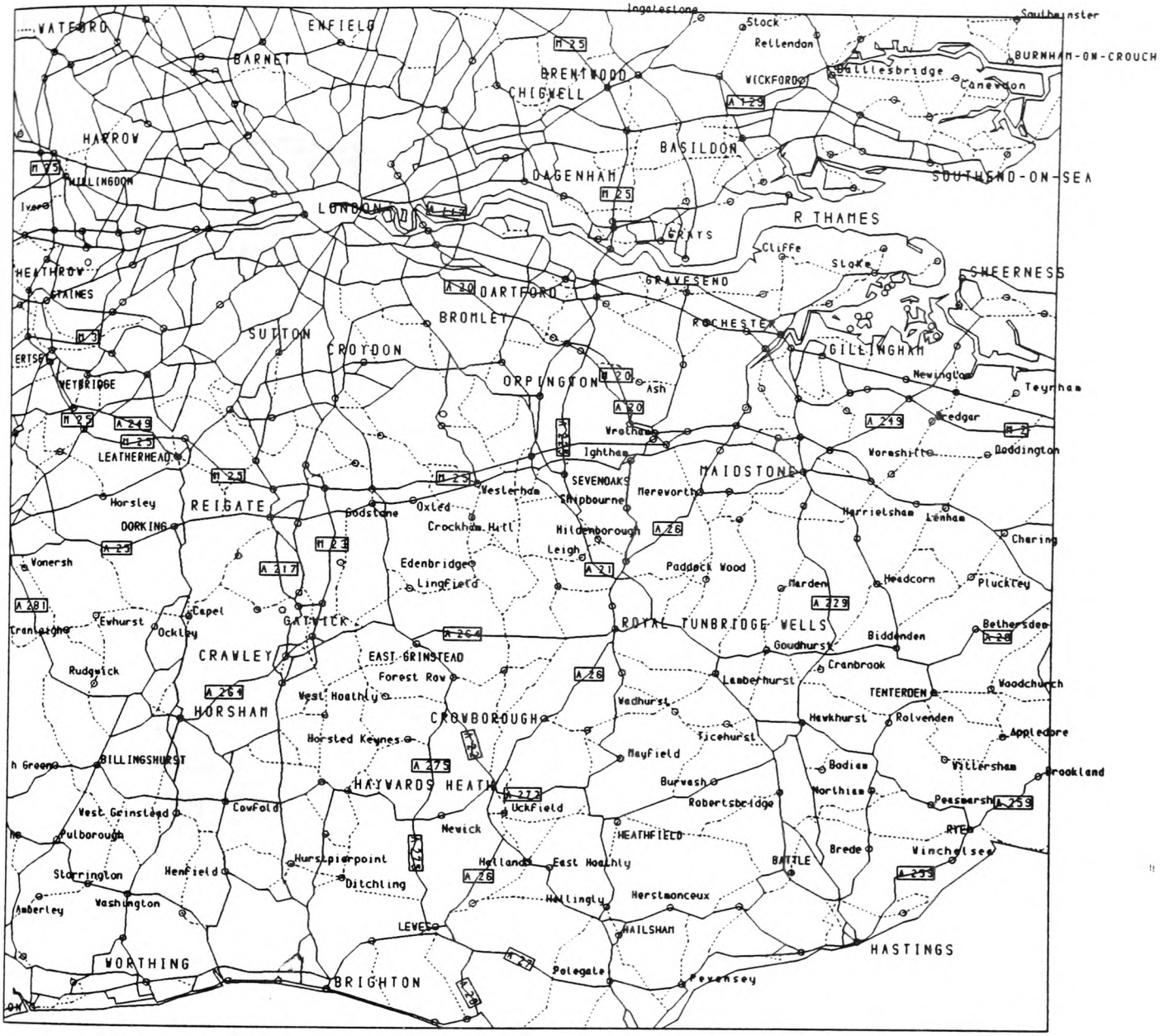
[8.30C] No town on a primary route to be placed within 5km of a city or 4km of a large town.

[8.30D] No large town to be placed within 4km of a city.

Because of the large number of labels in the Bristol area, B road labels were removed and the minimum separation distances between labelled point features increased (Fig 8.25):

[8.31A] No small town/village to be placed within 13km of a city, 11km of a large town, 9km of a primary route town or 7km of a primary route village.

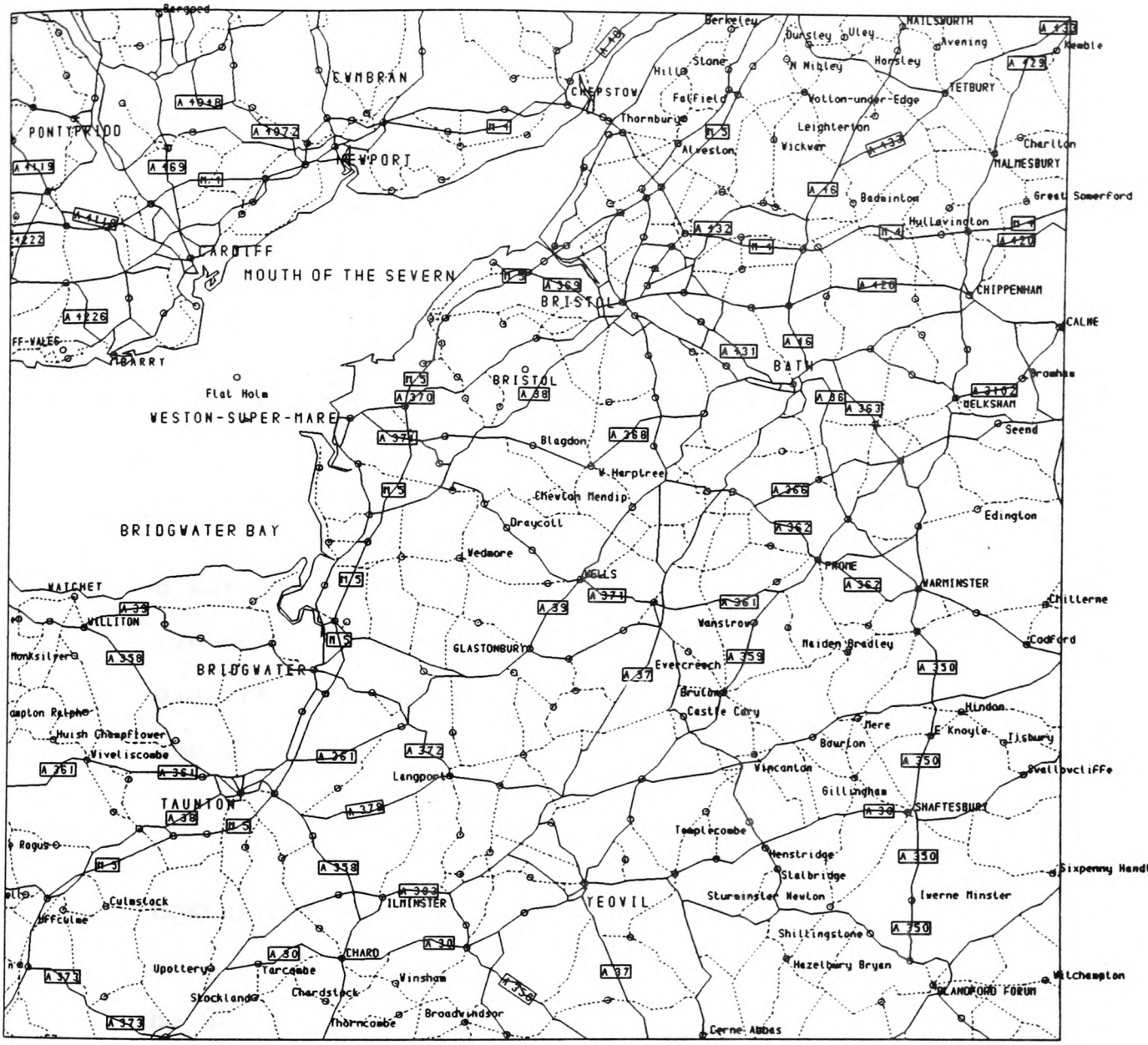
[8.31B] No village on a primary route to be placed within



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	500000	m
INORTH	Map Window	100000	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	500000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.24 Route Planner map of London - do not label large towns within 6km of cities etc.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	300000	m
INORTH	Map Window	100000	m
JEAST	Map Window	400000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	300000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	400000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.25 Route Planner map of Bristol region - no "B class" road labels and do not label large towns within 4km of cities etc.

11km of a city, 9km of a large town or 7km of a primary route town.

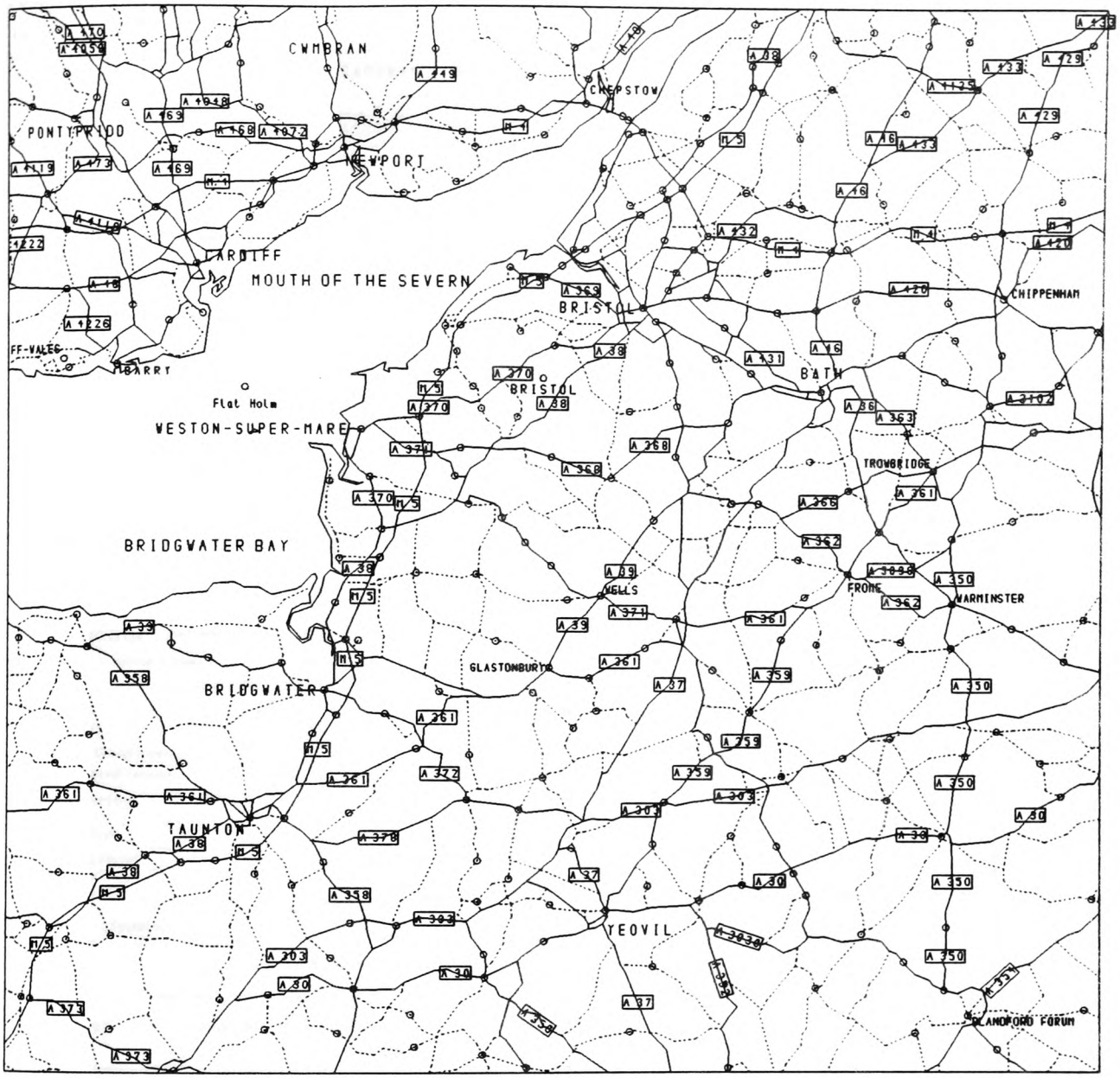
[8.31C] No town on a primary route to be placed within 9km of a city or 7km of a large town.

[8.31D] No large town to be placed within 7km of a city.

Unfortunately, this resulted in large areas of the region being swept of most settlement labels in Fig 8.25. So in Fig 8.26 and Fig 8.27 rules [8.29A] to [8.29D] were applied but modifications were made to emphasize roads (Fig 8.26-no villages or B roads labelled and A road label threshold reduced to 5km) and settlements (Fig 8.27-no road labels).

8.3.7 COMPARISON WITH LABPOS

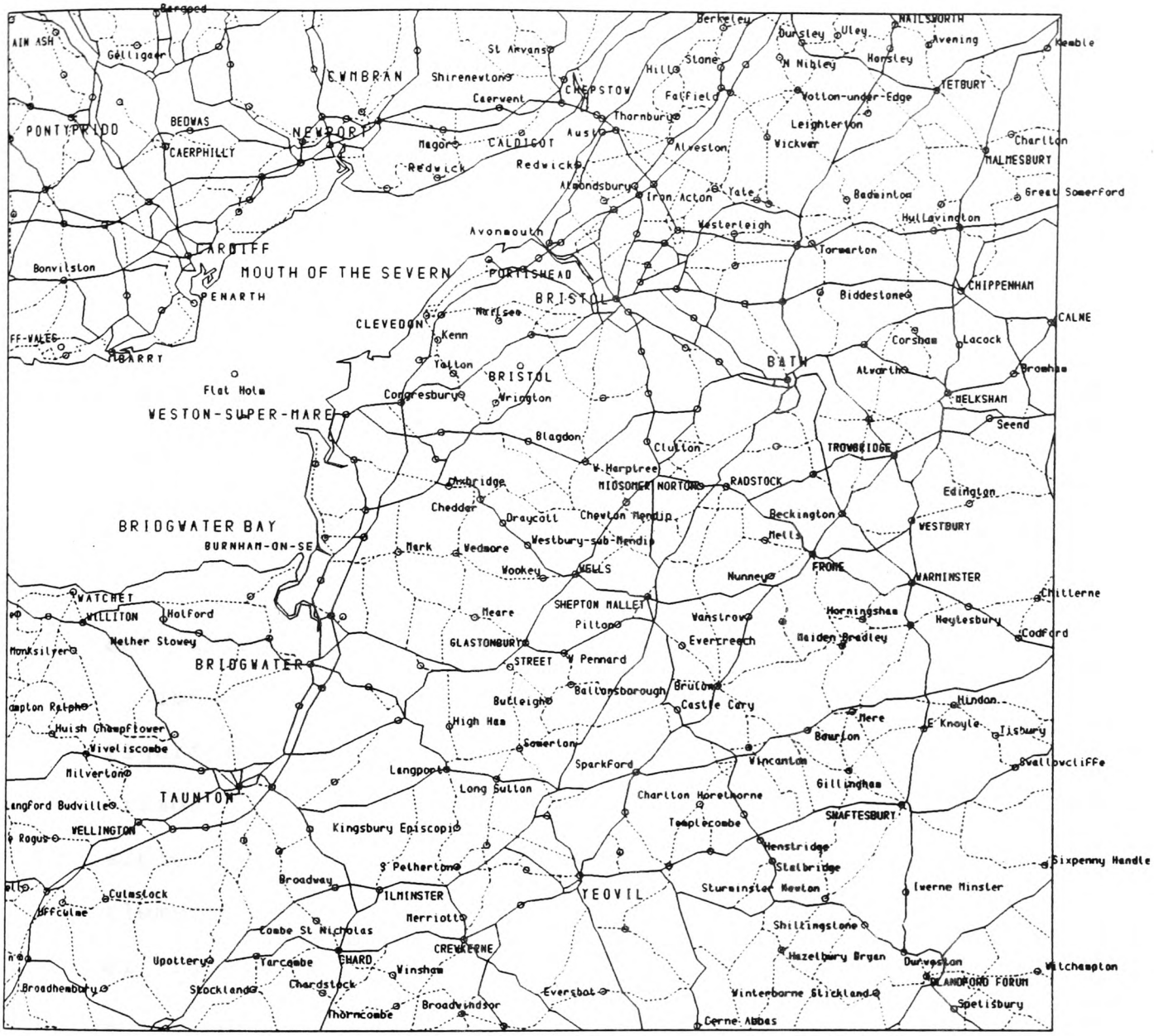
The use of the PROLOG strategy for placing labels on the Route Planner map differs from LABPOS in several respects. The ability to modify and include new rules exceeds the capability of LABPOS with its limited parameter files. However label placement was found to be typically a factor of ten slower in NAMEX, but still an order of magnitude faster than manual placement. The appearance of the Route Planner map is similar on both systems except for the fact that horizontal line label



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	300000	m
INORTH	Map Window	100000	m
JEAST	Map Window	400000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	300000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	400000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.26 Route Planner map of Bristol region - no village labels and do not label large towns within 4km of cities etc.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	300000	m
INORTH	Map Window	100000	m
JEAST	Map Window	400000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	350	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	300000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	400000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.27 Route Planner map of Bristol region - no road labels and do not label villages within 4km of large towns etc.

placement and area label placement does not take place in LABPOS and label splitting is not implemented on the NAMEX system. Also label deletion is used by the NAMEX strategy as an alternative to placing labels in conflict with other labels, as in LABPOS.

8.3.8 COMMENTS

This section of the chapter (Section 8.3) has shown the versatility of the NAMEX system for being able to change or add rules so as to affect the appearance of the Route Planner map. For instance it has been possible to select which roads to label according to their length or the importance of their junctions. Road label configuration can be selected according to the angle of the roads and local feature density. It has even been possible to increase the size of labels belonging to settlements lying near to the coast and to vary point label radius of proximity in feature dense regions of the map.

Although this example has been shown to work for several regions of the country, it does have some disadvantages. From the user's point of view, unintelligent deletion is a problem (Plymouth in Fig 8.15). From the computing point of view, the main weakness lies in the list length which occupies large amounts of work space during recursive list processing.

8.4 ADMINISTRATIVE AREA MAP

8.4.1 INTRODUCTION

This example is intended to demonstrate the placement of county area and city point labels. The area labels will be allowed to vary in size according to the size of the area feature they represent. To determine an area label size it is necessary to find which configuration of area label to use, horizontal or diagonal. Once this is known the appropriate label size can be selected. This is achieved by attempting to place the biggest possible representation of that label first and if this is not completely placed inside the area, to reduce the label size in steps until it is.

8.4.2 MAP SPECIFICATION

The Administrative Area map in this example consists of city point settlements, coast and county boundary lines and county areas. These are all extracted from the Ordnance Survey Route Planner map database via the DB_GENERATE (Section 7.1.3).

Like the first example, the map scale and window descriptions must be defined in the "WIND_DEF" file (Table 8.9). However because the regions covered in this example

Table 8.9 Map window definition, "WIND_DEF", file contents for the Administrative Area map (Southern England).

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Table 8.10
Cartographic features - "FEAT_DEF" file (excluding empty contents of the records) for the Route Planner map.

FCODE	DESCRIPTION	WIDTH (m)	BIT
110	coast	1000	1
113	breakwater	1000	1
162	county_boundary	1000	2
172	county	0	5
361	city	5000	3

Table 8.11 Raster classified feature groups - contents of the "RAST_DEF" file (excluding empty records) for the Administrative Area map.

BIT PLANE	DESCRIPTION	PRIORITY (x10)
0	rub_out	999
1	coast_n_bounds	5
2	county	10
3	city	999

Table 8.12 Parameterized text definition rule-base file, "TEXT_PARAM", contents (excluding empty records and fields) for the Administrative Area map.

FEATURE CODE	RADIUS PROX.	PREFERRED ORDER OF PLACEMENT POSITIONS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	MIN FONT NO.	FONT NO.	MAX FONT NO.
172	3000	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	2	3	4
361	1200	15	6	0	4	0	5	0	3	0	4	9	5	3	3	0	6	0	6	5	15	0	1	0

FEATURE CODE	LETTER MIN SEPARATION	LETTER SEP.	LETTER MAX SEPARATION	WORD MIN SEPARATION	WORD SEPARATION	WORD MAX SEPARATION	FEATURE TYPE
172	0	50	100	1500	2000	2500	2
361	0	0	0	0	1600	0	0

* KEY TO FEATURE CODES

172 = county 361 = city

Table 8.13 "FONT" characteristics for the Administrative Area map.

FONT NO.	BLOCK HEIGHT	BLOCK WIDTH	LETTER HEIGHT	LETTER WIDTH	LOWER CASE HEIGHT	LETTER DESCENDER HEIGHT
1	2200	1600	1900	1300	1000	200
2	2500	2000	2300	1750	1300	300
3	3000	2500	2800	2250	1700	400
4	3500	3000	3250	2700	2500	500

are larger, labels and features are bigger, and less underlying map detail is present than in section 8.3, it is appropriate to lower the pixel scale to one pixel per kilometre. This reduces memory requirements and speeds up the program.

Valid feature codes and associated descriptive names, widths and bit plane numbers must be defined in file "FEAT_DEF" (Table 8.10). The "RAST_DEF" file contents (Table 8.11) must also be defined in order to specify the feature class names and bit plane priorities (See Section 7.5).

Although not required for the above data conversion process, the "TEXT_PARAM" (Table 8.12) and "FONT" (Table 8.13) files are needed to specify which classes of feature can be labelled and what label positions, configurations and fonts are permitted.

8.4.3 LABEL SELECTION AND INITIALISATION

The label selection process is very similar to that in the previous Route Planner map example (Section 8.3.3), except that label validation is replaced by "find suitable label dimensions". In fact "find suitable label dimensions" is dual purpose since it also performs the role of label validation.

```
select_labels:-
   loop for all names,
      select label configuration,
      /* get suitable label dimensions & is label
         O.K. to place ? */
      find suitable label dimensions,
      write label record.
```

8.4.3.1 LABEL CONFIGURATION SELECTION

Label configuration selection is identical to the previous Route Planner map example except that only point (Rule [8.1]) and area label configurations (Rules [8.6], [8.7] and [8.8]) are selected.

8.4.3.2 FINDING SUITABLE LABEL DIMENSIONS [8.32]

The role of computing label dimensions and validation is performed by "find_suitable_label_dimensions". This consists of four parts, the first deals with just point labels, the remainder concerns the testing of three different sized area labels defined by the maximum, typical and minimum recommended fonts specified in the text definition rule-base. Doing things in this order ensures the biggest possible label will be placed inside the area concerned. Three input variables are used, these

are the feature type, the feature code and the "NAME_DEF" pointer.

```
/* For point labels only */
find_suitable_label_dimensions(0,Fcode,Npoint):-
   /* Recommended radius of proximity, font, letter and
      word separations */
   read_text_param(Fcode,[1,6,9,12],
     [Prox,Font,Let_sep,Word_sep]),
   valid_font(Font), /* Section 7.6.5 */
   /* Update label record with recommended values */
   write_label_details([6,12,13,14],
     [Prox,Font,Let_sep,Word_sep]),
   compute_label_dimensions(Font,Let_sep,
     Word_sep,0,Label_len),
   /* Write computed label length to "LABEL" record */
   write_label_details([5],[Label_len),
   /* Validate label (Section 8.4.3.3) */
   valid_label(Npoint), !.

/* For biggest possible area label */
find_suitable_label_dimensions(2,Fcode,Npoint):-
   read_text_param(Fcode,[7,10,13],
     [Max_font,Max_let_sep,Max_word_sep]),
   determine_label_dimensions(Npoint,Max_font,
     Max_let_sep,Max_word_sep),!.
```

```
/*    For    medium    area    label    size    */
find_suitable_label_dimensions(2,Fcode,Npoint):-
      read_text_param(Fcode,[6,9,12],[Font,
     Let_sep,Word_sep]),
      determine_label_dimensions(Npoint,Font,
     Let_sep,Word_sep), !.

/* For smallest area label */
find_suitable_label_dimensions(2,Fcode,Npoint):-
   read_text_param(Fcode,[5,8,11],
     [Min_font,Min_let_sep,Min_word_sep]),
   determine_label_dimensions(Npoint,Min_font,
     Min_let_sep,Min_word_sep),!.

/* Computes the area label dimensions, writes these
   to the label record and validates the label */
determine_label_dimensions(Npoint,Font,Let_sep,Word_sep):-
   valid_font(Font),

write_label_details([12,13,14],[Font,Let_sep,Word_sep]),
   compute_label_dimensions(Font,Let_sep,Word_sep,
     0,Label_length),
   /* Write label length to "LABEL" record */
   write_label_details([5],[Label_length]),
   valid_label(Npoint), !.
```

8.4.3.3 LABEL VALIDATION

"find_suitable_label_dimensions" makes use of the predicate "valid_label" to verify that the label is valid according to the criteria below. This is one of the three predicates defined in NAMEX, which should have one of its clauses, in this case "name_select", defined in LOGIC. Two variables are used, the "NAME_DEF" record number and feature type. The "name_select" component of the "valid_label" predicate called is similar to the previous map example in that all point labels are valid [8.1]. However no line labels are valid and area labels are defined thus:

[8.33] To avoid the problem of trivially sized coastal islands being given county names a minimum area size is specified so that if a label is an area label AND the area is larger than 250km sq in size THEN it is VALID:

```
name_select(No,2):-
     read_name_def_details(No,[3],[Fsn]),
     get_area_area(Fsn,Area),
     convert(1,pixels,Area_m,m),
     Area_km_sq is Area_m*Area_m*Area/1000000,
     Area_km_sq > 250,
     compute_current_label_dimensions,
     /* These positions are automatically fed
        into FORTRAN memory */
     determine_area_label_positions(Num),
     /* The area label has at least one
        position and so is valid */
     Num > 0, !.
```

8.4.4 VALIDATION OF LABEL POSITIONS

The validation of label positions is very similar to the Route Planner map example except that "test_larger_label" and "pos_test" (Section 8.3.4) are defined differently.

In the Administrative Area map, "test_larger_label" should prevent labels being placed too close to cities on the map.

[8.34] The minimum separation distance between a label and an adjacent city feature is defined using the radius of

proximity value in the parameterized text definition rule-base and the rule below. The minimum separation distance is 1.1 times the radius of proximity (In the LOGIC program this is is not strictly true since the vertical separation distance for area labels is "0.5") for a label, based upon the results from section 6.9.2.6.

```
test_larger_label(Lab_east,Lab_north,Length,Height,Prox):-
   Enlg_len:= Length+2.2*Prox,
   Enlg_ht:= Height+2.2*Prox,
   rasterize_rectangle_under_label(Enlg_len,Enlg_ht).
```

[8.35] IF the label represents a point or area feature and is too near a city THEN that position is INVALID and an empty list is returned.

```
   /* This relies upon the pixel contents found with
      "test_larger_label" */
   pos_test(_,_,_,[]):-
      rast_def(Plane,city,_),
      read_from_memory(Value,Plane),
      Value > 0, !.
```

As before, label positions are classified as lying in "free space", "dense space" or over too much underlying detail. In this example, "free space" is defined as no

underlying detail beneath a label position and "dense space" is defined as before (Section 8.3.4) except that a 50% threshold is used.

```
dense_space_threshold(50).
```

[8.36] IF the position tested has a ratio which is greater than the dense space threshold THEN the position is INVALID and an empty list is returned.

```
pos_test(Ratio,_,_,[]):-
   dense_space_threshold(Thrsh),
   Ratio > Thrsh, !.
```

[8.37] IF the position has a ratio less than or equal to the dense space threshold, THEN the position is valid and the weight and position are returned.

```
pos_test(Ratio,Pos,Weight,[Ans]):-
   dense_space_threshold(Thrsh),
   Ratio =< Thrsh,
      free_space_allowance(Res,Weight,Ans).

/* Free space */
free_space_allowance([0.0,Pos],Weight,[Weight,Pos]):- !.
```

```
/*Dense space */
free_space_allowance([Ratio,Pos],_,[Ratio2,Pos]):-
   Ratio > 0,
   Ratio2 is 0.0-Ratio, !.
```

8.4.5 MINIMUM SEPARATION BETWEEN LABELS

This is defined identically to the maximum label separation in the previous example (Section 8.3.5).

8.4.6 EXAMPLES

Three areas of the country have been used to demonstrate the capability of using NAMEX, namely southern England, central England, and Kent. Occasionally county area labels do not appear on the maps e.g. "ESSEX" in Fig 8.28. This occurs towards the map edges and may be due to problems in extracting county boundaries from the Ordnance Survey Route Planner database.

8.4.6.1 DEFAULT SETTINGS

Because of the smaller number of labels to be placed on the Administrative Area map, all 20 label positions are utilised (Fig 8.28 and Fig 8.29).

8.4.6.2 ALLOWANCE FOR COUNTY BOUNDARY AND COASTLINE [8.38]

Rule [2.27] recommends placing a point label, near a linear feature, on the same side of the line as the point. To test if a point label is on the wrong side of a county boundary, a rectangle is generated whose opposite corners lie on the centre of the label and the point feature coordinates. If the label centre should happen to lie across or entirely on the wrong side of county border, then the contents of this rectangle will include some county boundary pixels (Fig 8.30).

However, it is permissible for the label to lie on the wrong side of a county boundary if the boundary is part of a coastline (Rule [2.22]). To make an allowance for this, label positions are also tested to see if they contain any coastline and if so the presence of county boundary lines can be ignored (Fig 8.31 and Fig 8.32). The test to see if a label is on the wrong side of a county boundary, "wrong_side_of_county_test", is called from a modified version of "pos_test" (Rule [8.37]).

```
/* Not applicable for an area Label */
   wrong_side_of_county_test(Ans,[Ans]):-
      read_label_details([4],[2]), !.
```

```
/* A point label */
   wrong_side_of_county_test(In,Ans):-
      get_label_coords(Least,Lnorth),
      get_point_coords(Peast,Pnorth),
      Base:= abs(Peast-Least),
      Height:= abs(Pnorth-Lnorth),
      /* Test map space between label centre and point */
      rast_rect_init(East,North,Base,Height,0),
      rast_def(Plane1,coast_n_bounds,_),
      rast_def(Plane2,county,_),
      /* Check for coastline */
      read_from_memory(Value1,Plane1),
      /* Check for county boundary */
      read_from_memory(Value2,Plane2),
      boundary_test(Value1,Value2,In,Ans), !.

   /* Coastline found */
   boundary_test(Value1,_,Ans,[Ans]):-
      Value1 > 0, !.

   /* No coast or county boundary */
   boundary_test(0,0,Ans,[Ans]):- !.

   /* County boundary found but no coastline */
   boundary_test(0,Value2,Ans,[]):-
      Value2 > 0, !.
```

8.4.6.3 FILTER MINIMUM ALLOWABLE AREA SIZE

To show the need for a minimum allowable area size, Fig 8.33 shows what happens if the minimum area size is made too small (10km sq in this case).

8.4.6.4 DIFFERENT ANGLES AND ELONGATIONS FOR AREA LABELS

The use of diagonal or horizontal area labels can be favoured by adjusting the elongation and angle requirements in rules [8.6] to [8.8].

Fig 8.34 emphasizes diagonal area labels and was produced by setting the elongation requirement to 0.2 and the angle limits to +/-5 degrees. Fig 8.35 was produced with the elongation set to 0.7 and the angles limits at +/-20 degrees and clearly emphasizes horizontal area labels.

8.4.7 SECTION SUMMARY AND DISCUSSION

This section of the chapter has demonstrated the ability of the NAMEX system to cope with area label placement and area orientated rules. One of these rules involved fitting the biggest possible label, out of a range of three sizes, into a county area. In order to

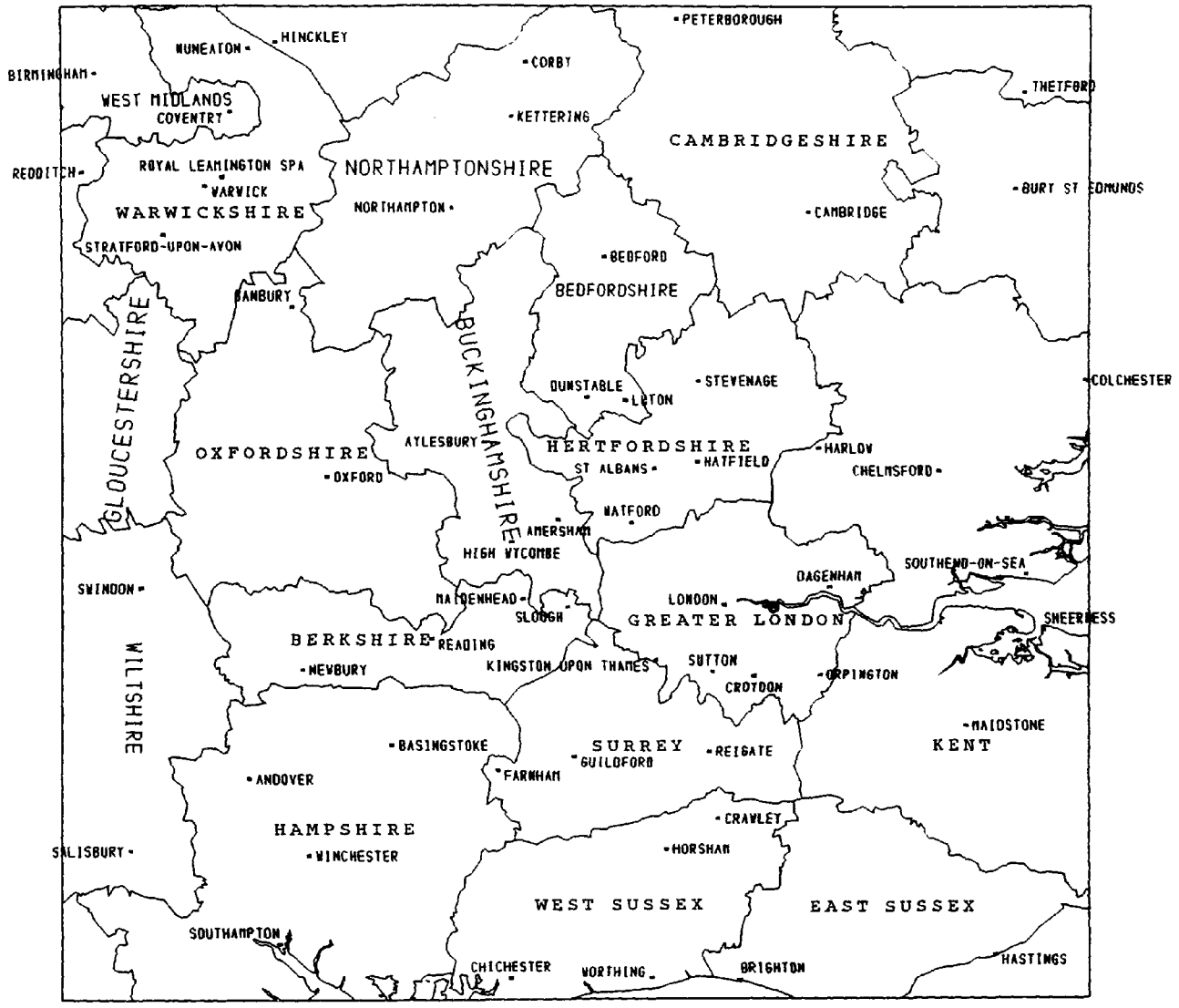
avoid every island belonging to a mainland county from being labelled with the county name, a minimum area threshold was introduced. A means of avoiding placing a settlement label on the opposite side of a county boundary to the settlement was successfully implemented. A demonstration of how selected label configurations for county areas can easily be altered, by changing the specified elongation and angle of the area for the configurations concerned, is given.

Unfortunately although many of the above rules generally appear to work on most labelled features some errors were observed. For instance in all of the maps showing mainland "KENT", it appears that a larger "KENT" label could equally well fit inside the county. This is especially noticeable in Fig 8.33 where the island label "KENT" is bigger than the mainland "KENT". Also, although ambiguity between horizontal county area labels and cities was supposed to be avoided, this actually occurs, for instance "BERKSHIRE" and "READING" in Fig 8.29. The reasons for these discrepancies remains unclear. However the problem of selecting the wrong area label sizes may have something to do with problems the author experienced with extracting complete county boundary data from the source Ordnance Survey Route Planner map database.

Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.28 Default Administrative Area map of southern England.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	100000	m
JEAST	Map Window	600000	m
JNORTH	Map Window	300000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	300000	m
SCALE	Map Scale	625000	

Fig 8.29 Default Administrative Area map of central England.

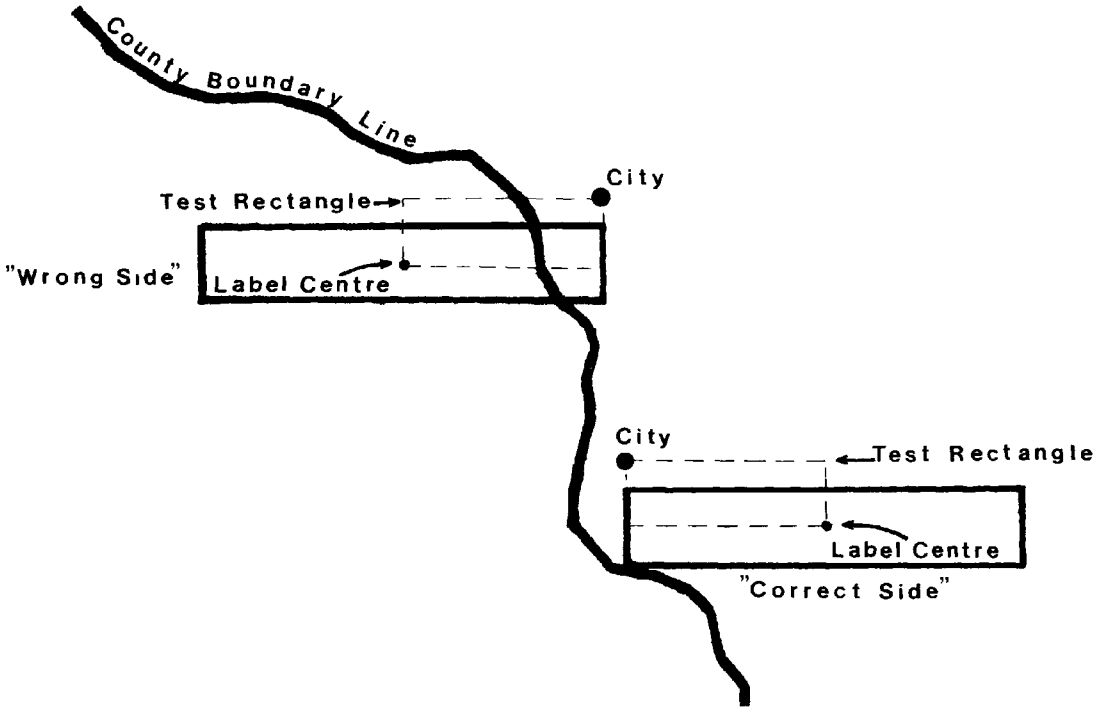
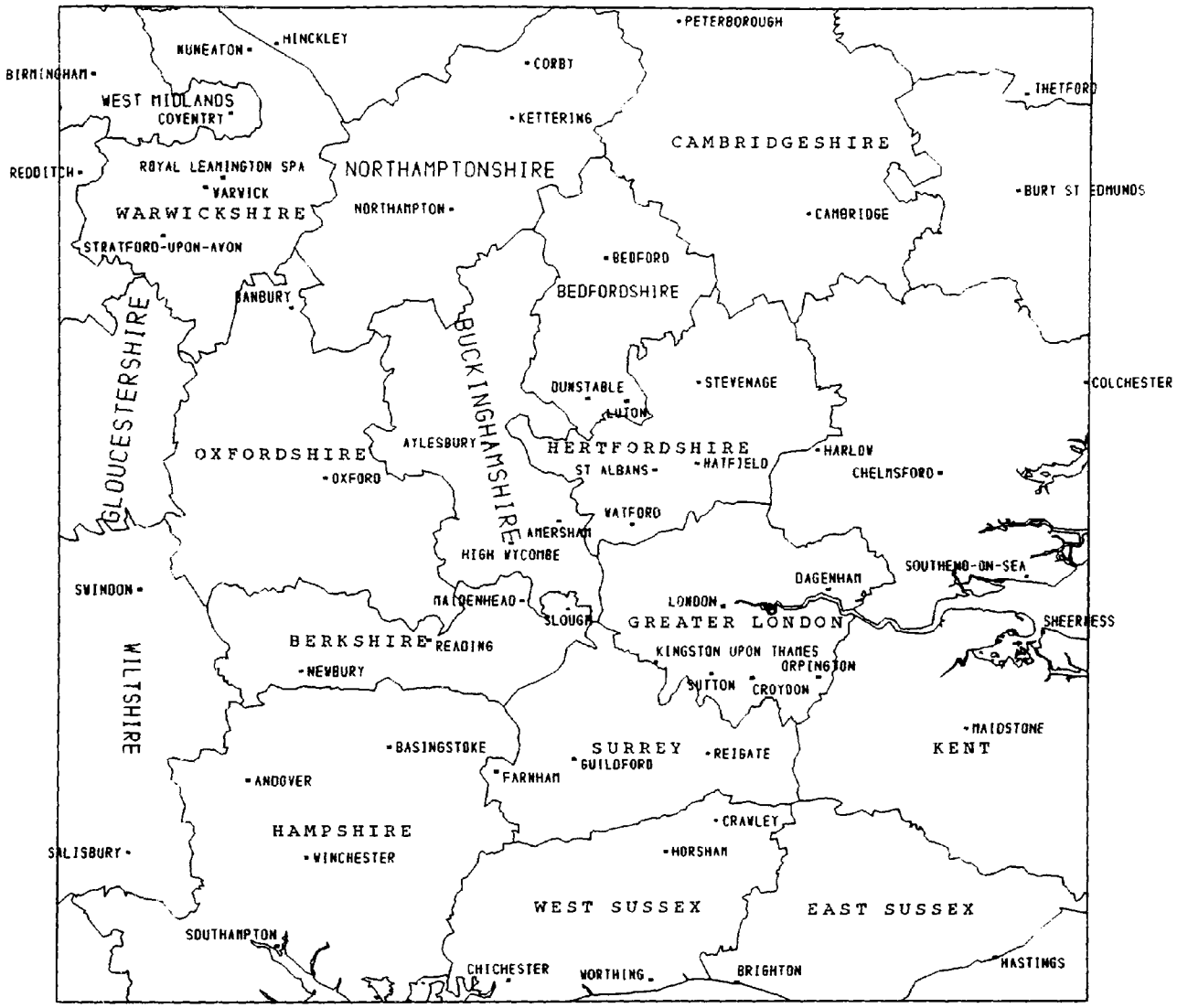


Fig 8.30 Label is on the wrong side of the county boundary to the city if the test rectangle contains county boundary pixels.

Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.31 Administrative Area map of southern England after taking into account the wrong side of the boundary test.



Window Definition File "WIND_DEF" contents

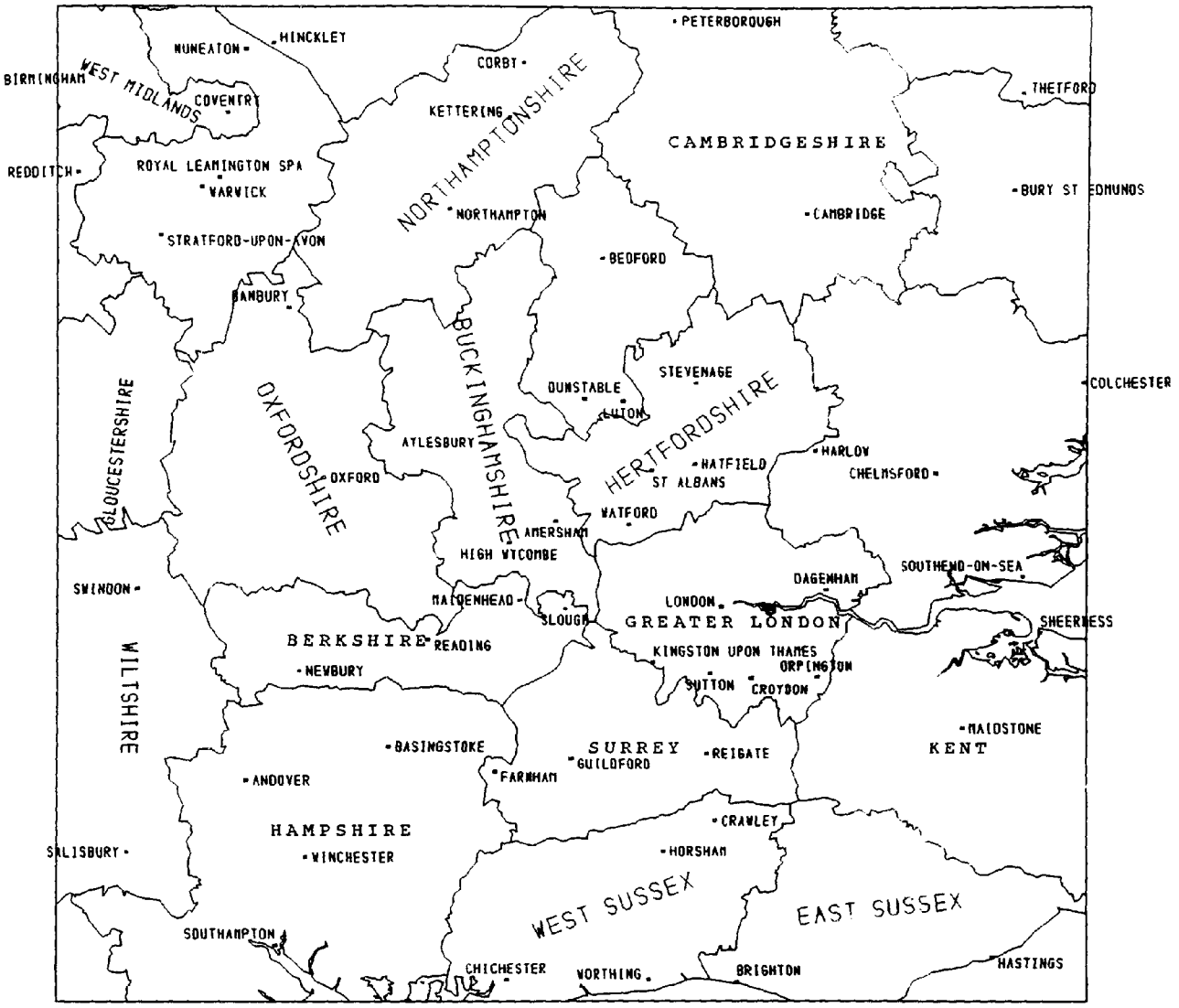
RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	100000	m
JEAST	Map Window	600000	m
JNORTH	Map Window	300000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	300000	m
SCALE	Map Scale	625000	

Fig 8.32 Administrative Area map of central England after taking into account the wrong side of the boundary test.

Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	600000	m
INORTH	Map Window	100000	m
JEAST	Map Window	700000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	600000	m
LNORTH	Raster Window	100000	m
MEAST	Raster Window	700000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

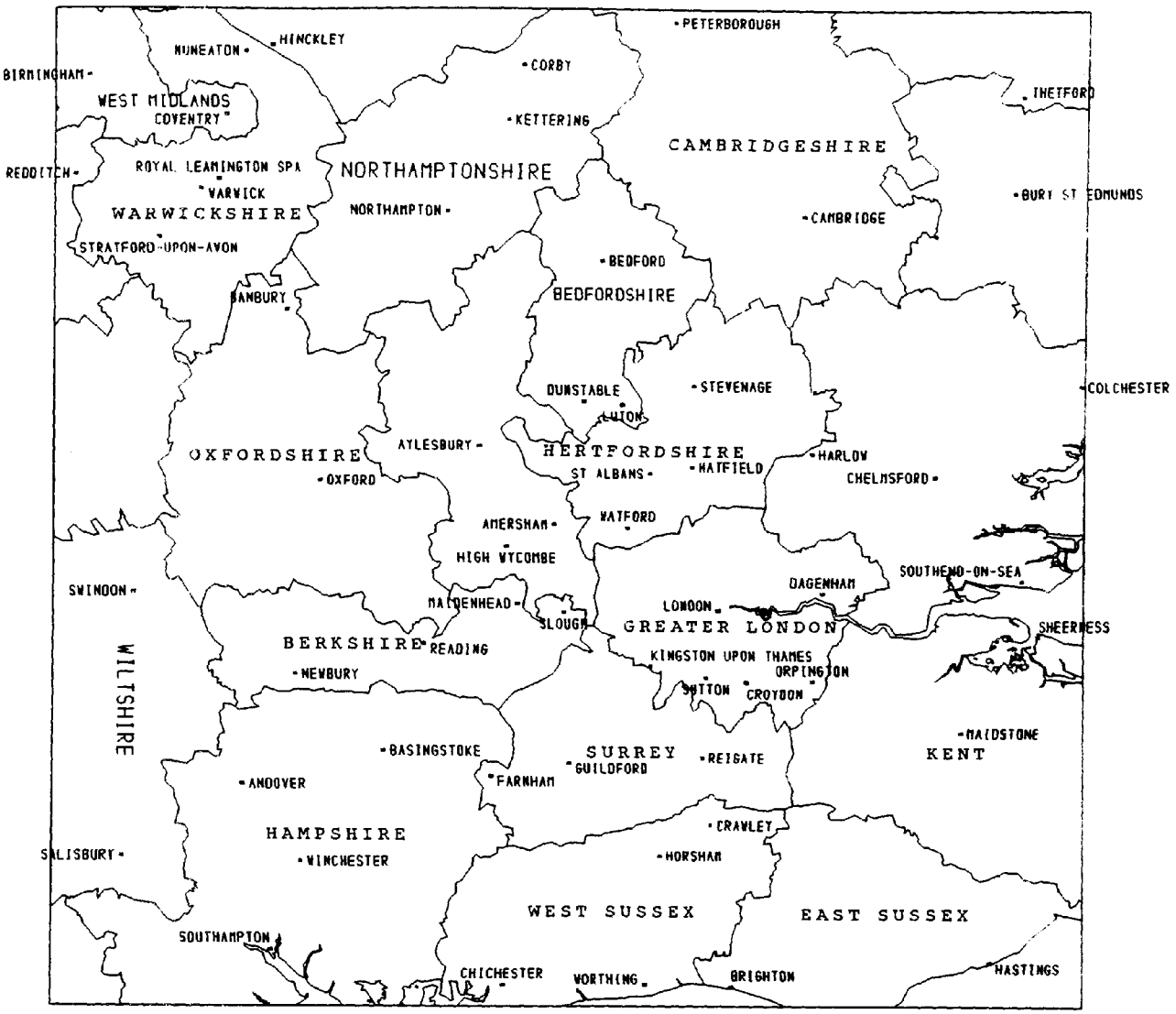
Fig 8.33 Administrative Area map of Kent with the minimum allowable area size set at 10km sq.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.34 Administrative Area map of central England with a preference for diagonal area labels.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	400000	m
INORTH	Map Window	0	m
JEAST	Map Window	600000	m
JNORTH	Map Window	200000	m
IUNIT	Raster Size	100	pixels
ISIZE	Grid Sq. Size	100000	m
LEAST	Raster Window	400000	m
LNORTH	Raster Window	0	m
MEAST	Raster Window	600000	m
MNORTH	Raster Window	200000	m
SCALE	Map Scale	625000	

Fig 8.35 Administrative Area map of central England with a preference for horizontal area labels.

8.5 MOON MAP

8.5.1 INTRODUCTION

The most common features present on Moon maps are craters which can be approximated to ellipses. Due to the variation in size of craters, the small ones can be treated as point features whilst others may be large enough to completely contain labels and can be regarded as areal features. Associated with the primary (main) craters are several secondary (minor) craters whose labels can be abbreviated to just their suffix letter over a specified range of map scales (Section 8.5.3.3 and Fig 8.46).

Unlike the previous two examples, generalisation will be used so as to allow the Moon map to be produced at different scales. Also the conventional point and area positioning system is replaced by a system which allows for changes in label size and configuration with position number.

8.5.2 MAP SPECIFICATION

The cartographic vector and raster data for the Moon map are extracted via the DB_GENERATE program from a file of named lunar craters. This consists of the names, latitudes, longitudes and diameters of 8498 named lunar craters (ANDERSSONand WHITAKER).

To cater for varied representation of craters, the NAMEX database allows each crater to be stored as both point and area features. The raster bit planes are divided into five classes: primary craters, secondary craters, regions beyond the Moon's limb, primary point-like craters and secondary point-like craters. These are defined in the "RAST_DEF" file (Table 8.14). There are just two feature codes used: primary crater and secondary crater, these are defined in the "FEAT_DEF" file (Table 8.15).

Like the first example, the map window specifications and raster size must be defined in the "WIND_DEF" file (Table 8.16). The projection system used for the map is orthographic, and the Moon can be viewed from any angle as defined by the DB_GENERATE program. The number of craters plotted varies with the square of the scale (See Rule [8.42]), but only half of these craters are considered for labelling since the maps produced show only one hemisphere. On Moon maps studied by the author, secondary crater labels did not appear until the map scale became larger than 1:7,500,000. Because of this it was decided to exclude all secondary craters from being plotted or labelled until the scale became larger than 1:7,500,000.

Although not required for the above data conversion process, the "FONT" file is needed to specify the font characteristics. The character fonts have a range of sizes

Table 8.14 Raster classified feature groups - contents of the "RAST_DEF" file (excluding empty records) for the Moon map.

BIT PLANE	DESCRIPTION	PRIORITY (x10)
0	rub_out	999
1	primary	15
2	secondary	10
3	beyond_limb	40
4	primary_point	1
5	secondary_point	1

Table 8.15 Cartographic features - contents of the "FEAT_DEF" file (excluding empty records) for the Moon map.

FCODE	DESCRIPTION	WIDTH (m)	BIT
0	primary_crater	0	1
1	secondary_crater	0	2

Table 8.16 Map window definition, "WIND_DEF", file contents for 1:24,000,000 scale Moon map.

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	80000	m
INORTH	Map Window	80000	m
JEAST	Map Window	3920000	m
JNORTH	Map Window	3920000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	3840000	m
LEAST	Raster Window	80000	m
LNORTH	Raster Window	80000	m
MEAST	Raster Window	3920000	m
MNORTH	Raster Window	3920000	m
SCALE	Map Scale	24000000	

which cover all practical scales at which the Moon map will be produced and increase in dimensions by approximately half for each step (Table 8.17). Unlike the previous two map examples, the available positions and configurations are defined in rules in the LOGIC program so the parameterized text definition rule-base file, "TEXT_PARAM", is not needed for this example.

8.5.3 LABEL SELECTION

The label selection process is similar to the first map example (Section 8.3.3) except that a check is made to see if the crater is valid to label (valid crater) and the label text may need to be abbreviated (process text) depending upon the crater type and the map scale. The label selection process is illustrated in the logic pseudo code below:

```
select_labels:-
   loop for all names,
      valid crater, /* Is crater O.K. to label ? */
      select label configuration,
      process text, /* i.e. abbreviation ? */
      write label record.
```

Table 8.17 "FONT" definition file for the Moon map.

FONT NO.	BLOCK HEIGHT	BLOCK WIDTH	LETTER HEIGHT	LETTER WIDTH	LOWER CASE HEIGHT	LETTER DESCENDER HEIGHT
1	10	7	8	5	1	1
2	16	11	13	8	2	2
3	25	17	20	13	3	3
4	40	27	32	21	4	4
5	63	42	50	34	6	6
6	100	67	80	53	10	10
7	158	106	127	85	16	16
8	251	167	201	134	25	25
9	398	265	318	212	40	40
10	631	421	505	337	63	63
11	1000	667	800	533	100	100
12	1585	1057	1268	845	158	158
13	2512	1675	2010	1340	251	251
14	3981	2654	3185	2123	398	398
15	6310	4206	5048	3365	631	631
16	10000	6667	8000	5333	1000	1000
17	15849	10566	12679	8453	1585	1585
18	25119	16746	20095	13397	2512	2512
19	39811	26540	31849	21232	3981	3981
20	63096	42064	50477	33651	6310	6310
21	100000	66667	80000	53333	10000	10000
22	158489	105660	126791	84528	15849	15849
23	251189	167459	200951	133967	25119	25119
24	398107	265405	318486	212324	39811	39811

8.5.3.1 CRATER SELECTION

Because this example covers a wide range of map scales, it is not possible to display or label all craters. Therefore crater selection criteria must be established for different ranges of scales. This is performed using "valid_crater(Fcode,Sca,Fsn)":

[8.39] If the crater is a primary crater (Fcode=0) and satisfies the generalisation rule [8.42] then it is valid to label.

```
valid_crater(0,Sca,Fsn):-
    crater_diameter(Diameter,Fsn),
    generalise(Sca,Diameter).
```

[8.40] If the crater is a secondary crater (Fcode=1) and the scale is larger than 1:7,500,000 and smaller than or equal to 1:3,000,000 and satisfies the generalisation rule [8.42] and it is an external crater (Not inside a primary crater) then it is valid to label.

```
valid_crater(1,Sca,Fsn):-
    Sca >= 3000000,
    Sca < 7500000,
    generalise(Sca,Diameter),
    external_crater(Fsn).
```

[8.41] If the crater is a secondary crater (Fcode=1) and the scale is smaller than 1:3,000,000 then it is valid to label.

```
valid_crater(1,Sca,_):-
    Sca < 3000000.
```

On manually produced Moon maps, craters are usually selected for plotting or labelling according to their visibility. The visibility of a crater varies with its brightness, contrast with surrounding map detail and diameter. Unfortunately information was only available on the latter in the named crater data file so the crater selection criteria used in this example depends upon diameter only.

On the Moon's surface, the number of craters increases with a decrease in crater diameter. This is a consequence of the cratering history of the Moon and most of the other planetary bodies in the solar system (Guest and Greeley). By experimenting with the file of named crater data (ANDERSSONand WHITAKER), the author found an approximate relationship between a threshold crater diameter D and the number of craters N on the Moon's surface (both hemispheres) larger than this threshold diameter (Fig 8.36).

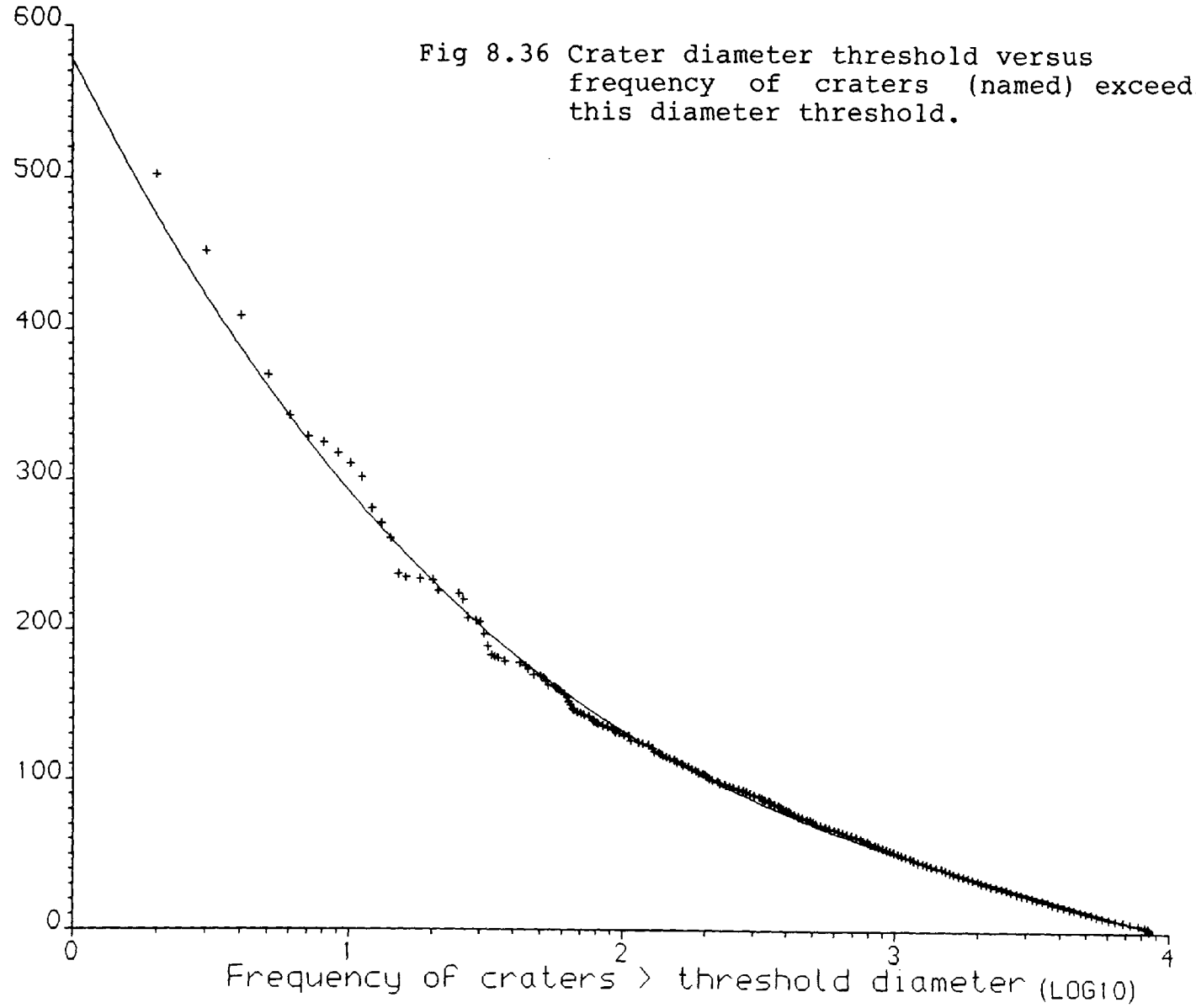


Fig 8.36 Crater diameter threshold versus frequency of craters (named) exceeding this diameter threshold.

$$D=578.5-366.194(\text{Log } N)+88.3094(\text{Log } N)^2-8.2606726(\text{Log } N)^3$$

Where D = crater threshold diameter (km).

N = Number of craters on the Moon's surface larger than D.

[8.42] At a map scale of 1:15,000,000 there should be some 500 craters over the whole Moon which can be labelled (The smallest being 71km). Ideally as the scale changes the same number of labels should be placed per unit area on the map.

```
generalise(Sca,Diam):-
    L=Log((15000000/Sca)^2*500),
  Limit=1000*(578.5-366.194*L+88.3094*L^2-8.2606726*L^3,
  Diam >= Limit.
```

In practice however there will be considerably less. This is mainly due to an increase in apparent crater density towards the Moon's limb (a foreshortening effect which results in a larger surface area being presented) resulting in fewer label positions. If labels have few or no positions they will either not be selected in the first place or will be removed later on. Local variations in crater density and size distributions also occur over the Moon's surface, for instance the Moon's far side is more

densely cratered than the Earth facing or near side.

8.5.3.2 LABEL CONFIGURATION SELECTION

Two label configurations are catered for:

[8.43] If the elongation (eccentricity) of a crater is less than or equal to 0.55 then the label configuration is that of a "near circular" crater (Configuration = 11, Fig 8.37).

```
select_label_config(11):-
   get_area_elong(E),
   E =< 0.55.
```

[8.44] If the elongation (eccentricity) of a crater is greater than 0.55 then the configuration is that of an "elongated" crater (Configuration = 12, Fig 8.38).

```
select_label_config(12):-
   get_area_elong(E),
   E > 0.55.
```

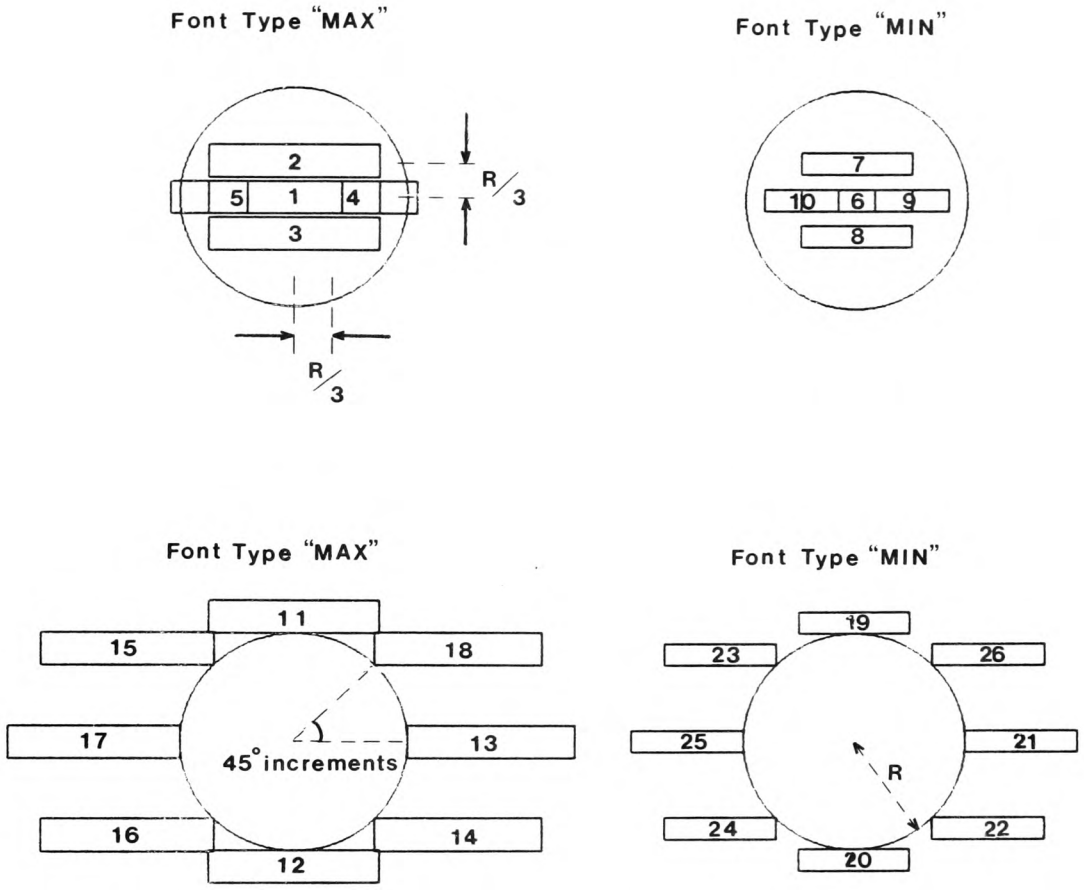


Fig 8.37 Near circular crater configurations and positions.

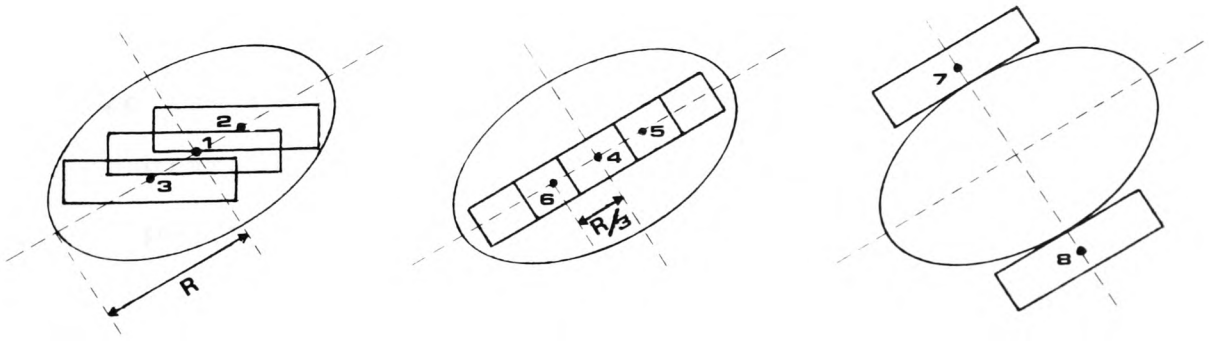


Fig 8.38 Elongated crater configurations and positions.

8.5.3.3 ABBREVIATION

Secondary crater labels can sometimes be abbreviated to their suffix letters. This is done using:

process_text(Fcode,Sca,Let_count,New_let_count).

Secondary craters can usually be identified because the last word consists of one or two letters. To abbreviate the label, it is necessary to search the second or third character from last in a label and if this is a space (A secondary crater label) the last letter(s) is rewritten at the start of the label and the new letter count is set to one (or two).

[8.45] Do not abbreviate primary crater labels.

```
process_text(0,Sca,Let_count,Let_count):-
   put_text_into_mem, !. /* FORTRAN memory */

/* For secondary craters: */
process_text(1,Sca,Let_count,New_let_count):-
   crater(Type),
   put_text_into_mem, /* FORTRAN memory */
   process_text2(Type,Sca,Let_count,New_let_count), !.
```

[8.46] If secondary crater is external to a primary crater and the scale is between 1:3,000,000 and 1:7,500,000 then abbreviate the label.

```
process_text2(external,Sca,Let_count,New_let_count):-
    Sca >=3000000,
    Sca < 7500000,
    abbreviate_text(Let_count,New_let_count). /* See
                                program documentation */
```

[8.47] If the secondary crater is external to a primary crater and the scale is larger than 1:3,000,000 do not abbreviate.

```
process_text2(external,Sca,Let_count,Let_count):-
    Sca < 3000000.
```

The remaining "process_text2" predicates are defined similarly:

[8.48] If the secondary crater is internal to a primary crater do not abbreviate or label unless the scale is smaller than or equal to 1:3,000,000.

[8.49] If the secondary crater is internal to a primary crater and the scale is between 1:1,000,000 and 1:3,000,000 abbreviate the label.

[8.50] If the secondary crater is internal to a primary crater and the scale is larger than 1:1,000,000 do not abbreviate.

8.5.4 GENERATION OF PREFERRED ORDER OF POSITION LISTS

This varies slightly from the method given in Section 8.2.5.2.1 to take into account the different positions/configurations available to crater labels. This is illustrated in the logic pseudo code below:

```
generate_position_order:-
   loop for all labels,
   get_label_details,([Fsn,Config]),
   positions_available(Config,No_of_pos),
   mask_out_current_feature, /* Area */
   compose_pos_list(No_of_pos,[],Lst),
   sort_and_extract_weighted_position_list(Lst,Poslist),
   asserta(placement_order(Label,Poslist)).

/* Near circular crater configuration, 26 positions */
positions_available(11,26).

/* Elongated crater configuration, 8 positions */
positions_available(12,8).
```

"compose_pos_list" is similar to that in section

8.2.5.2.1, however the priority is defined as the negated inverse of the position number. This will favour the lower number positions when the position list is sorted.

8.5.4.1 VALIDITY OF LABEL POSITIONS

"valid_position" (Section 8.2.5.2.1) is used to check label positions and determine their preference. The returned preference is either the weight determined by "compose_pos_list" (above), if the position lies in "free map space" else, the proportion of underlying detail covered by the label if it lies in "dense map space".

```
valid_position(Pos,Weight,Res):-
    read_label_details([3,22],[Fsn,Config]),

    /* Compute label details for given position */

find_label_state(Fsn,Pos,Labeast,Labnorth,Ang,Ln,Ht),
    mask_out_feature(Fsn,2),/* Mask out crater area */
    rast_rect_init(Labeast,Labnorth,Ln,Ht,Ang),
    window_pixel_sum(Pix_sum),/* underlying detail */
    window_pixel_area(Pix_area),/* label area */
    Ratio is 100*Pix_sum/Pix_area,
    pos_test(Ratio,Weight,Pos,Res).
```

"pos_test" has the job of returning a list to "valid_position". The list is empty if the position is invalid, but contains the position and its associated preference if it is valid.

[8.51] The free space threshold is assumed to be 0% throughout the program.

[8.52] The dense space threshold is 20%.

[8.53] Do not select a position if the label lies over too much underlying detail. Same as rule [8.15].

[8.54] If the label lies over the centre of a primary crater, the position is invalid.

```
pos_test(_,_,_,[]):-
    rast_def(Plane,primary_point,_),
    read_from_memory(Value,Plane),
    Value > 0.
```

[8.55] If the label lies inside its own crater (Fig 8.37 and Fig 8.38), then it is only valid if at least three quarters of it lies inside the crater area. Label positions lying outside the crater are always valid. This rule is used in rules [8.56] and [8.57] below.

```
pos_test(_,_,Pos,[]):-
   not(pos_test2(Pos)).

/* External near circular crater configuration labels */
pos_test2(Pos):-
   read_label_details([22],[11]),
   Pos >= 11.

/* External elongated crater configuration labels */
pos_test2(Pos):-
   read_label_details([22],[12]),
   Pos >= 7.

/* All internal crater labels */
pos_test2(Pos):-
   /* Remove mask to examine the labelled crater */
   erase_mask,
   read_label_details([3],[Fsn]),
   /* Compute label details for given position */
   find_label_state(Fsn,Pos,Labeast,Labnorth,Ang,Ln,Ht),
   window_pixel_area(Rect_area), /* Label area */
   rast_free_pix(Free_pix), /* Free space pixels */
   Ratio is (Rect_area-free_pix)/Rect_area,
   /* At least 3/4 of label inside its own crater */
   Ratio > 0.75.
```

[8.56] If the label lies in free map space and satisfies rule [8.55] then the weight returned is that provided by "compose_pos_list" (Section 8.5.4).

```
pos_test(Ratio,Weight,Pos,[[Weight,Pos]]):-
    Ratio = 0, /* Free map space */
    pos_test2(Pos).
```

[8.57] If the label lies in dense map space and satisfies rule [8.55] then the weight returned is the ratio of underlying detail to label area provided by "compose_pos_list" (Section 8.5.4).

```
pos_test(Ratio,_,Pos,[[Ratio,Pos]]):-
    Ratio > 0.0, /* Dense map space */
    dense_space_threshold(Threshold),
    pos_test2(Pos),
    Ratio =< Threshold.
```

8.5.4.3 LABEL POSITIONS

Unlike the previous map examples (Sections 8.3 and 8.4), the label position system is defined by a set of PROLOG predicates:

"find_label_state(Fsn,Pos,Lab_east,Lab_north,Ang,Len,Ht)"

and

"pos(Config,Pos,East,North,Angle,Crater_diam,Font_type)"

(described in the separate program documentation). The "state" of a label refers to its coordinates, angle and size for a given position number. The "pos" predicate computes the label's position given the position and configuration numbers and the diameter of the crater. The position numbering system for near circular configuration labels goes from 1 to 26 (Fig 8.37) and can include internal and external placements and two font sizes: "min" and "max" (Section 8.5.6.3). Elongated crater configuration labels have 8 positions (Fig 8.38), but only two of these are external and the "max" font size is used throughout.

8.5.5 GENERATION OF POTENTIAL LABEL OVERLAP LISTS

This is performed differently from the method given in Section 8.2.5.2.2, in that each labelled crater is tested against each other labelled crater to see which crater pairs (centres of craters) lie within a computed threshold separation distance. Those that do are stored in the usual potential overlap lists. The threshold

separation distance is the sum of the radius of circle of label containment for both craters in a crater pair.

A circle of label containment is computed for each crater to give the approximate extent of the region which the label may occupy within the vicinity of each crater. In the case of near circular craters the circle of containment radius is the sum of the crater radius, the label length and a character width. In the case of elongated craters the radius of the circle of containment is the crater radius if this is larger than the half the label length, else half the label length if the circle radius is smaller than half the label length.

8.5.6 ANCILLARY LUNAR CRATER FACTS

8.5.6.1 CRATER DIAMETER

Crater diameters can be computed indirectly from the pixel area of the crater which is always an ellipse of known elongation (eccentricity):

area of ellipse A = π.a.b

now $b = a\sqrt{1-e^2}$

so

$$A = \pi a^2\sqrt{1-e^2}$$

Semi-major axis length

$$a = \frac{A}{\pi.\sqrt{1-e^2}}$$

So the crater diameter Dm = 2a

This is defined as follows:

```
crater_diameter(Dm,Fsn):-
    get_area_area(Fsn,Area),
    get_area_elong(Elong),
    Dpix is 2*sqrt(Area/(Pi*sqrt(1-Elong^2)),
    convert(Dpix,pixels,Dm,m).
```

8.5.6.2 EXTERNAL/INTERNAL CRATER

Secondary craters can be tested to see if they are internal or external to a primary crater with:

```
crater(internal):- not(external_crater).
crater(external):- external_crater.

external_crater(Fsn):-
    get_point_coords(Fsn,East,North),
    rast_point_init(East,North),
    rast_def(Plane,primary),_),
    read_from_memory(0,Plane), !.
```

"external_crater" is defined similarly but uses the current point coordinates.

8.5.6.3 FIND SUITABLE FONT

Twenty four character fonts are available in the "FONT" file and these are arranged in order of increasing size. Each crater has two sized fonts determined, "min" and "max", which are selected according to the label position number being used.

[8.58] The minimum letter size is 1.5mm for map scale concerned.

```
min_char_size(1.5,mm).
```

[8.59] The "minimum" letter size used should be as near to 1.5mm in height as is permitted.

```
find_suitable_font(Diam,Font,min):-
    min_char_size(Ht,mm),
    convert(Ht,mm,Min_ht), /* Convert to metres */
    loop for all fonts
        read_font(Height),
        Height >= Min_height.
```

[8.60] The "maximum" letter size used should ideally be of the order of 1/10th of crater diameter in size.

```
find_suitable_font(Diam,Font,max):-
    loop for all fonts
        read_font(Height),
        Height >= Diam/10.
```

8.5.6.4 COMPUTING RADIUS OF PROXIMITY

Because craters are usually elliptical in shape, the radius of proximity must be allowed to vary. This is achieved using:

```
compute_prox(Pos,Diam,Prox):-
    get_area_elong(E),
    get_area_orient(Ang1),
    ang(Pos,Ang2),
    A is Diam/2.0,
    B is A*sqrt(1-E*E),
    Bc is B*cos(Ang2-Ang1),
    As is A*sin(Ang2-Ang1),
    Prox is A*B*sqrt(1/(Bc*Bc+As*As)).

/* Position angles */
    ang(1,0.0).
    ang(3,45.0).
    ang(6,90.0).
    ang(9,135.0).
    ang(11,180.0).
    ang(13,225.0).
    ang(16,270.0).
    ang(19,315.0).
```

8.5.8 EXAMPLES

The following figures, Fig 8.40 to Fig 8.49, demonstrate the effects of generalisation of labelling on a Moon map as the scale is changed from 1:24,000,000 to 1:1,060,660 at intervals of root two. The footprints covered by the different scales are shown in Fig 8.39.

The smallest scale map, Fig 8.40, has a mixture of "elongated" and "near circular" crater label configurations. It is interesting to note that crater/label ambiguity is present near the bottom of the map. This is due to high feature density and often occurs on manually produced maps of the Moon as well. However because of the way that craters have been selected, according to size, the labels can be interpreted as belonging to the biggest, and usually closest, crater in their vicinity.

The map scale is large enough in Fig 8.41 for "near circular" crater names to start being placed internally (e.g. "LAMONT") and in Fig 8.42 "elongated" crater names start being placed internally (e.g. "BALMER"). "near circular" craters near the limb, such as "PETAVIUS" appear to be slightly misplaced. This may be due to errors in precision whilst computing the crater diameter from the crater pixel area (Section 8.5.6.1).

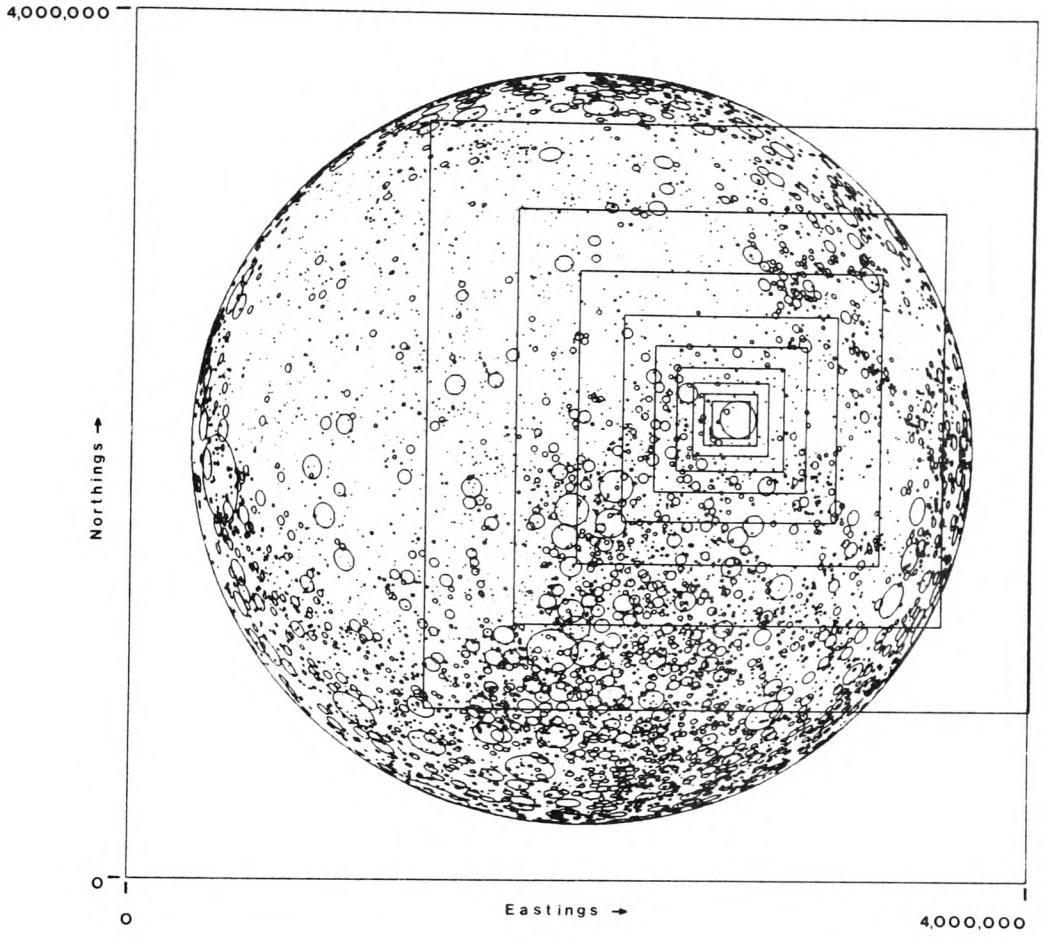


Fig 8.39 Foot prints on the Moon's surface of Figs 8.40 to 8.49.

A large number of craters are "switched in" when the scale becomes larger than 1:7,500,000 (Fig 8.44), this is a consequence of rules [8.40] and [8.41]. Abbreviation of secondary crater labels (Rules [8.46] and [8.49]) takes place at scales below 1:7,500,000 until Fig 8.47 is reached (scale larger than 1:3,000,000) when full names appear.

Primary crater names, such as "THEON SENIOR" in Fig 8.44, are allowed to be placed over secondary craters. This is a reasonable solution because there is no free space in which to place them. Also a very small number of craters, for some unknown reason have been given diagonal labels in Fig 8.44 and Fig 8.45.

Finally, to demonstrate the ability to view the Moon from any angle, Fig 8.50 was produced for the lunar far side.

8.5.9 SUMMARY AND COMMENTS

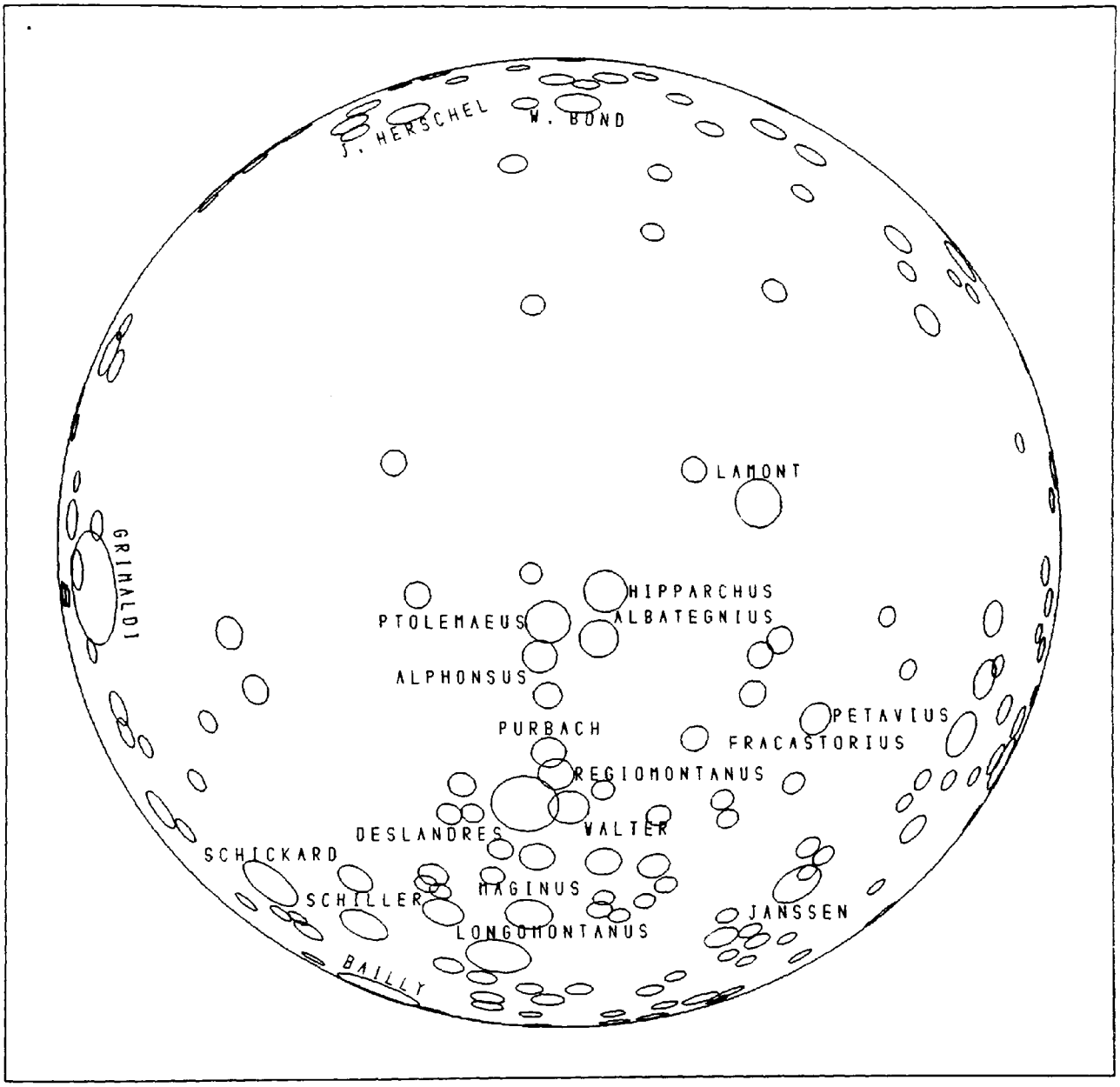
This section of the chapter has shown the flexibility of the rule-based approach to name placement. For instance it has been possible to override the standard rule-base for text definition and to build in name placement

positions and configurations which are different from those originally envisaged for use in the NAMEX system. The method of crater label overlap detection also differs from that used previously. Label abbreviation has also been successfully applied to secondary crater labels which are abbreviated between two scale limits.

The maps produced in this section have been encouraging because they have shown how generalisation can be built into a rule-based name placement system. The extremes of the map scales covered are just over twenty times. However upon reflection, the introduction of secondary craters at scales larger than 1:7,500,000 should be more gradual and perhaps secondary craters should be allowed at smaller scales providing that they satisfy the generalisation criterion (Rule [8.42]). Local crater density should also be taken into account so that the crater threshold diameter could be lowered or increased where appropriate.

However a small number of inconsistencies are present. These include "near circular" craters towards the Moon's limb having slightly misplaced labels due to precision errors in computing the radius of proximity. Also, very occasionally near circular craters have diagonally placed labels (Fig 8.44 and 8.45). Although it has not been possible to identify the cause of the latter problem, to avoid the effects of precision errors in

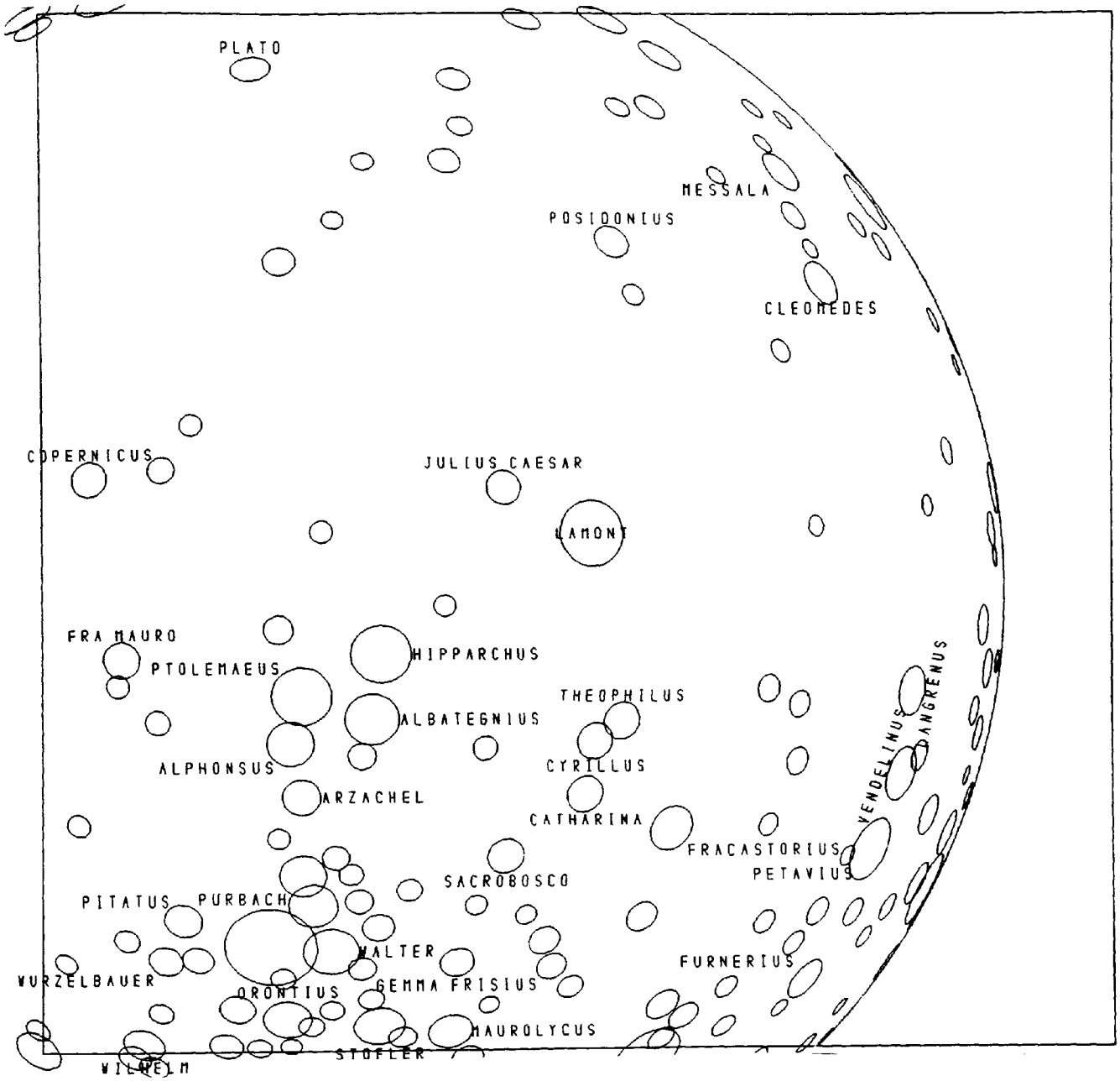
computing the radius of proximity (Section 6.5.8), the definition of "near circular" craters (Rule [8.43]) should be changed so that these only apply to craters with an elongation less than 0.5. The corresponding definition for "elongated" crater label configurations should also be changed.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	80000	m
INORTH	Map Window	80000	m
JEAST	Map Window	3920000	m
JNORTH	Map Window	3920000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	3840000	m
LEAST	Raster Window	80000	m
LNORTH	Raster Window	80000	m
MEAST	Raster Window	3920000	m
MNORTH	Raster Window	3920000	m
SCALE	Map Scale	24000000	

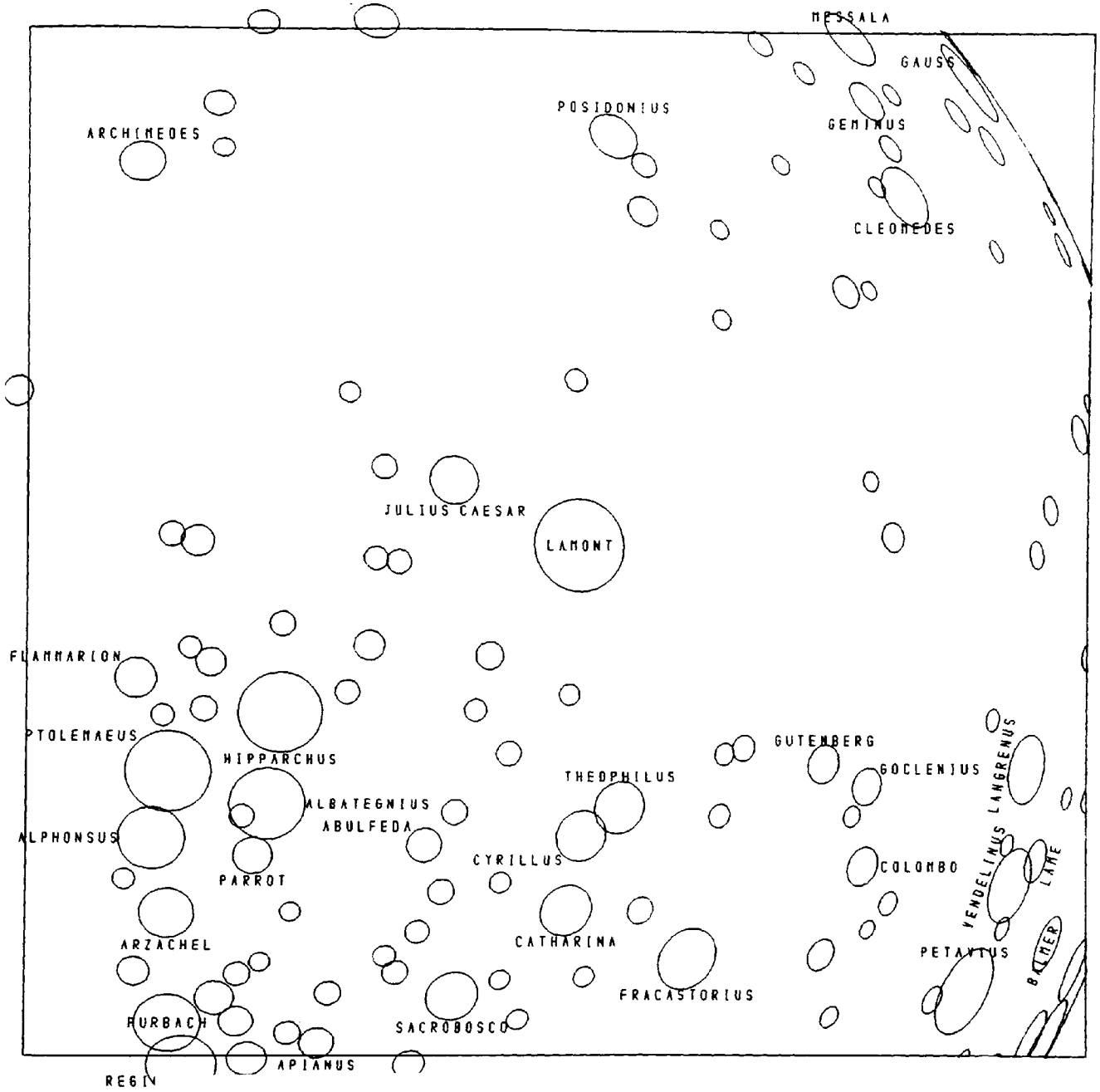
Fig 8.40 Moon's near side, scale 1:24,000,000.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	1292355	m
INORTH	Map Window	792355	m
JEAST	Map Window	4007645	m
JNORTH	Map Window	3507645	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	2715290	m
LEAST	Raster Window	1292355	m
LNORTH	Raster Window	792355	m
MEAST	Raster Window	4007645	m
MNORTH	Raster Window	3507645	m
SCALE	Map Scale	16970563	

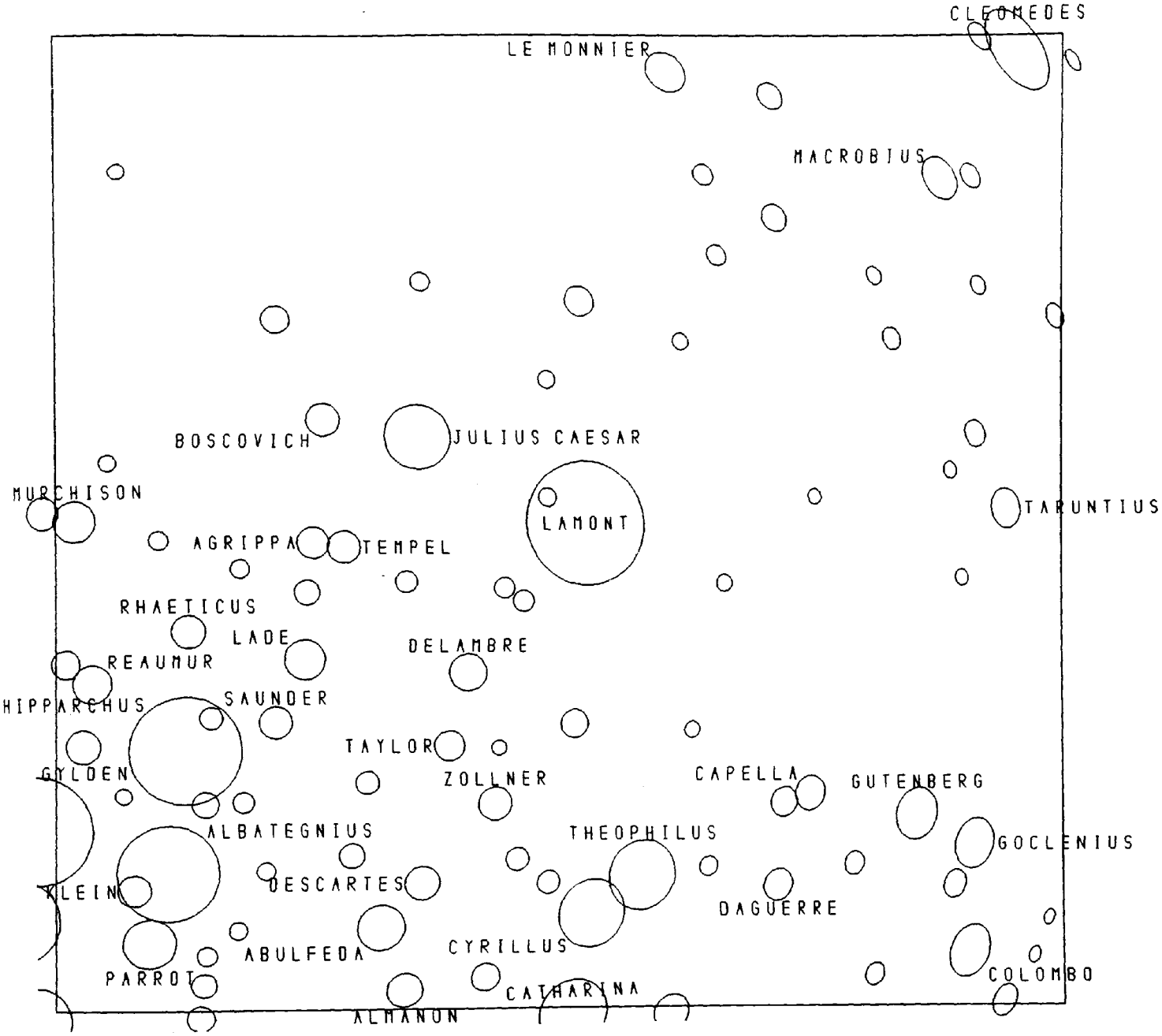
Fig 8.41 Moon's near side, scale 1:16,970,563.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	1690000	m
INORTH	Map Window	1190000	m
JEAST	Map Window	3610000	m
JNORTH	Map Window	3110000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	1920000	m
LEAST	Raster Window	1690000	m
LNORTH	Raster Window	1190000	m
MEAST	Raster Window	3610000	m
MNORTH	Raster Window	3110000	m
SCALE	Map Scale	12000000	

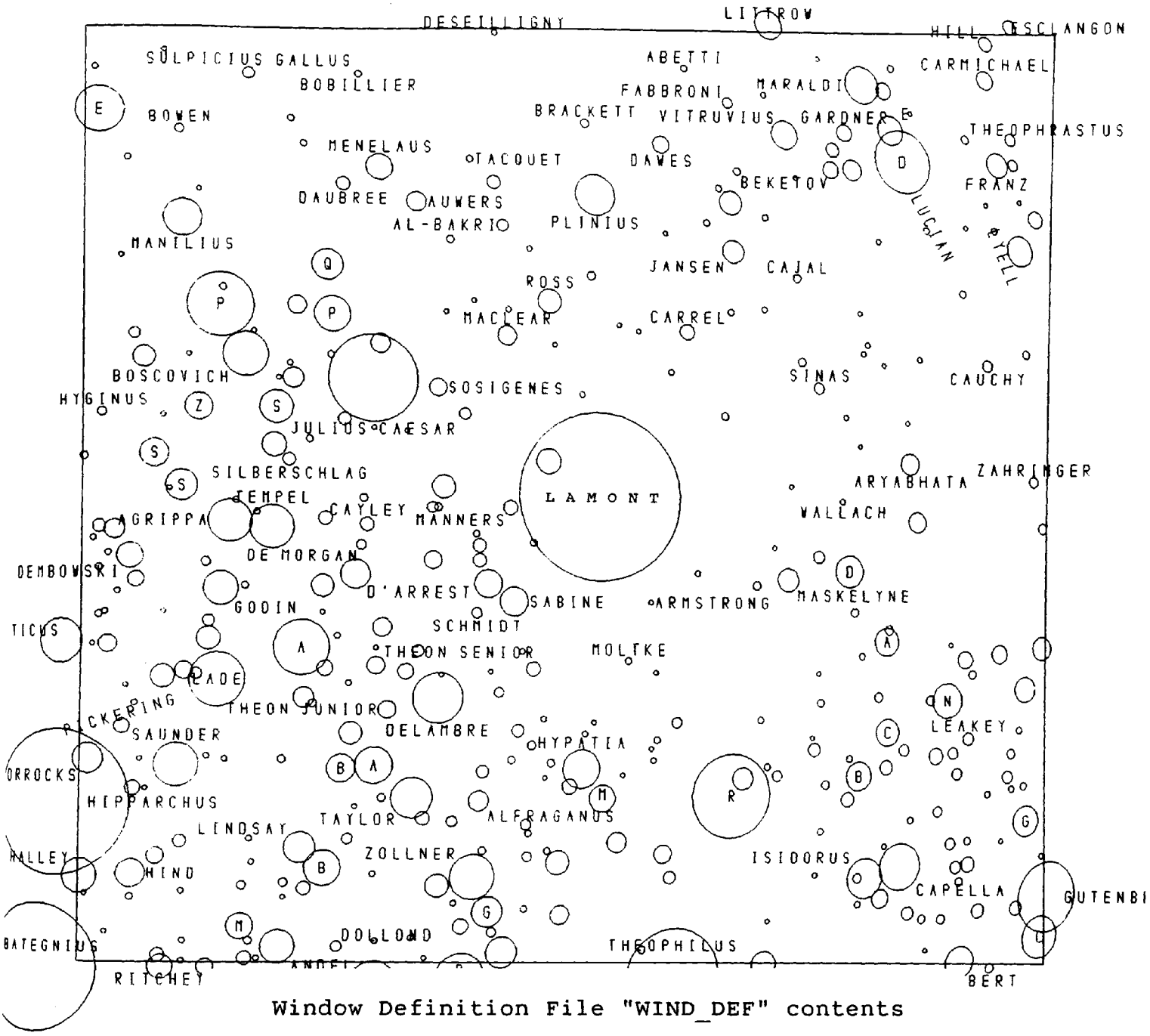
Fig 8.42 Moon's near side, scale 1:12,000,000.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	1971178	m
INORTH	Map Window	1471178	m
JEAST	Map Window	3328823	m
JNORTH	Map Window	2828823	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	1357645	m
LEAST	Raster Window	1971178	m
LNORTH	Raster Window	1471178	m
MEAST	Raster Window	3328823	m
MNORTH	Raster Window	2828823	m
SCALE	Map Scale	8485281	

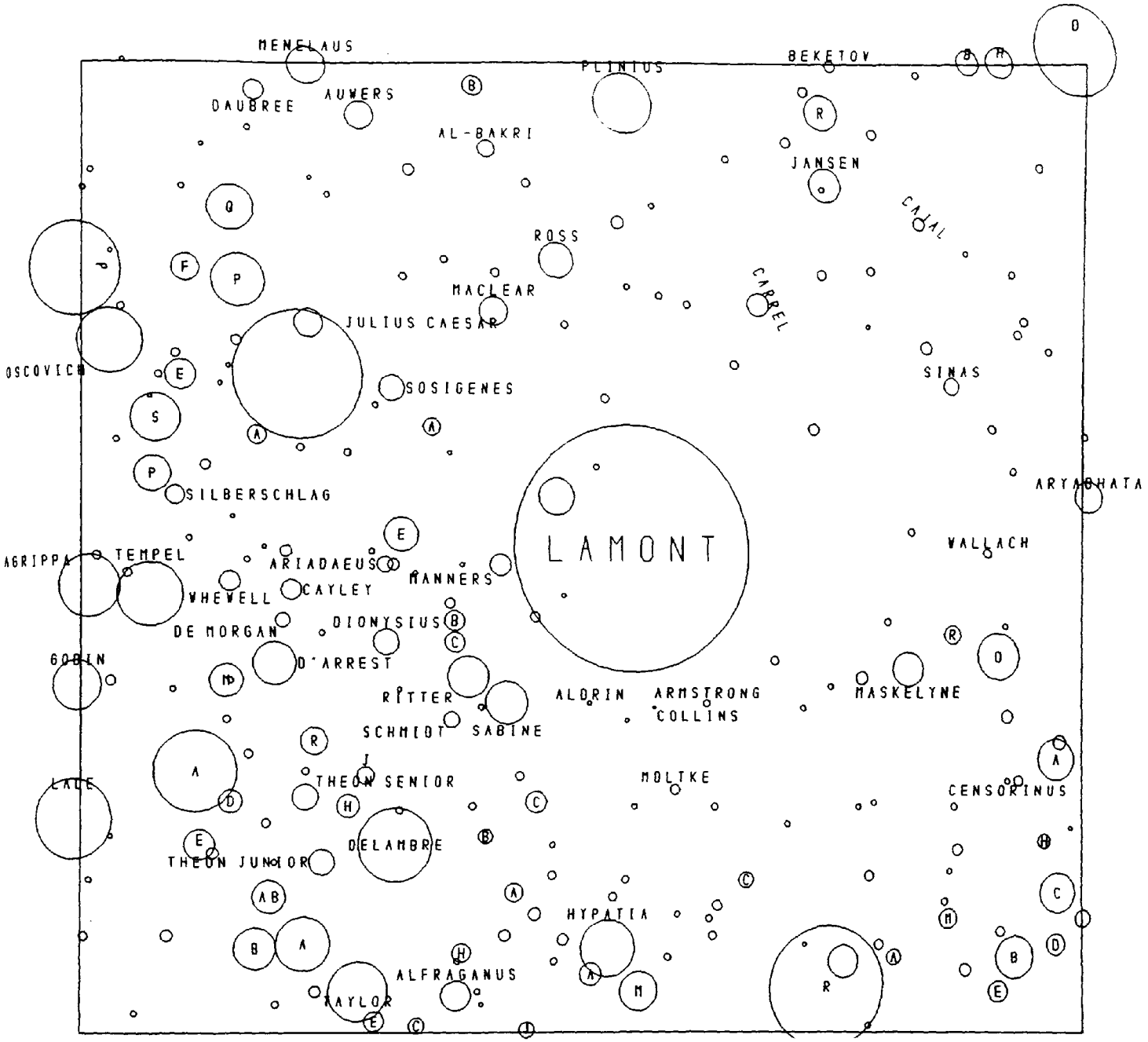
Fig 8.43 Moon's near side, scale 1:8,485,281.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	2170000	m
INORTH	Map Window	1670000	m
JEAST	Map Window	3130000	m
JNORTH	Map Window	2630000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	960000	m
LEAST	Raster Window	2170000	m
LNORTH	Raster Window	1670000	m
MEAST	Raster Window	3130000	m
MNORTH	Raster Window	2630000	m
SCALE	Map Scale	6000000	

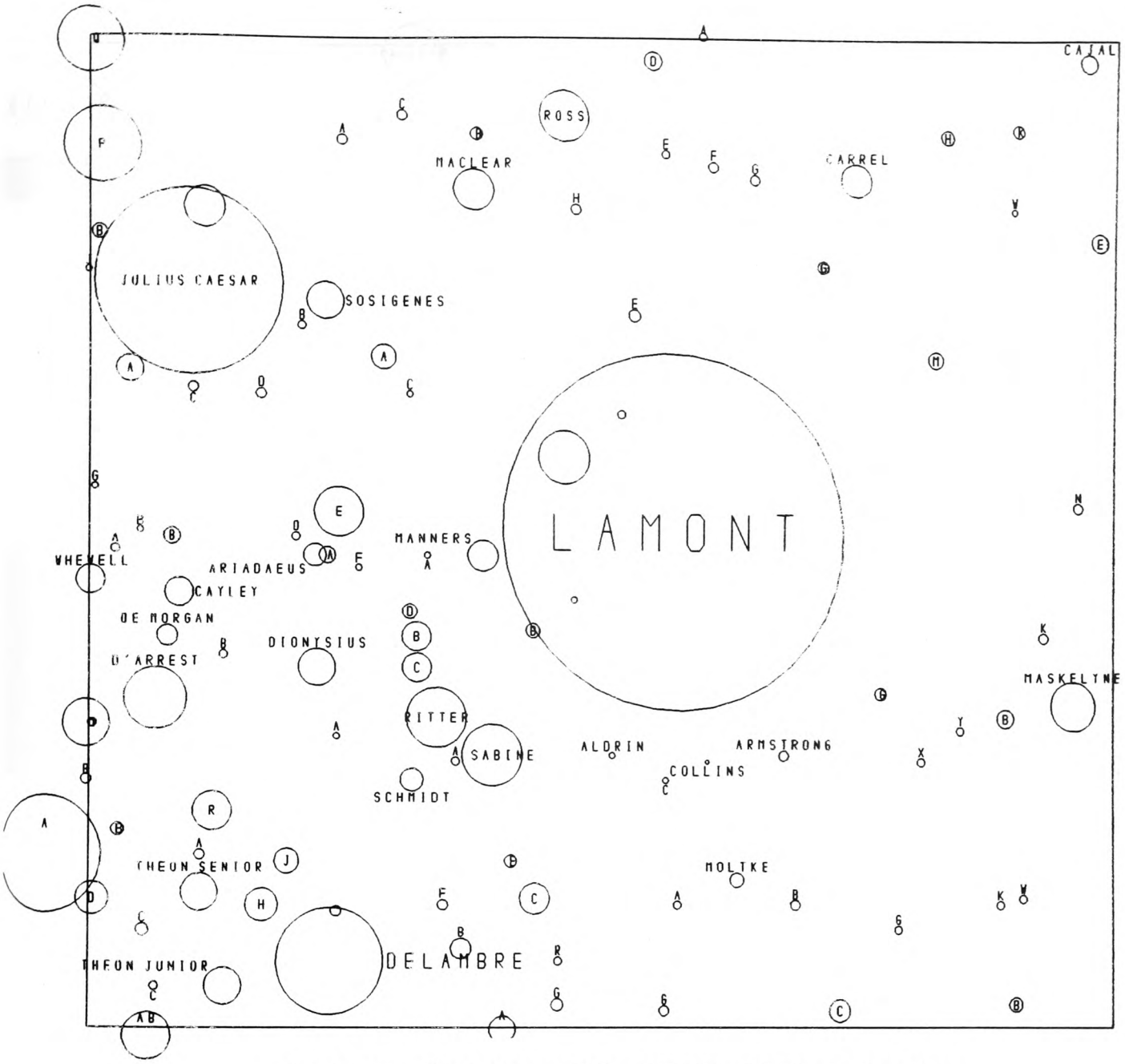
Fig 8.44 Moon's near side, scale 1:6,000,000.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	2310589	m
INORTH	Map Window	1810589	m
JEAST	Map Window	2989412	m
JNORTH	Map Window	2489412	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	678823	m
LEAST	Raster Window	2310589	m
LNORTH	Raster Window	1810589	m
MEAST	Raster Window	2989412	m
MNORTH	Raster Window	2489412	m
SCALE	Map Scale	4242641	

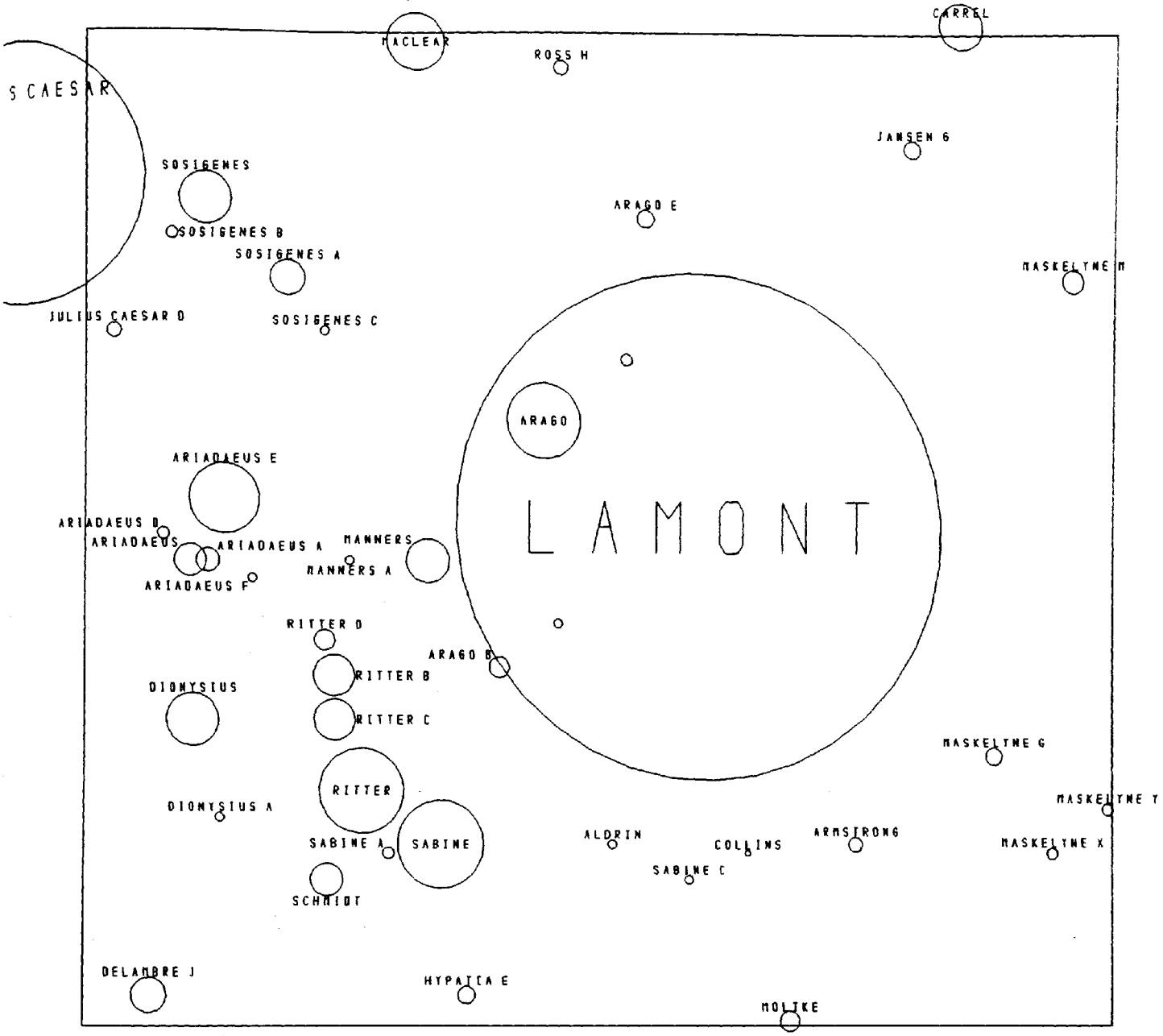
Fig 8.45 Moon's near side, scale 1:4,242,641.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	2410000	m
INORTH	Map Window	1910000	m
JEAST	Map Window	2890000	m
JNORTH	Map Window	2390000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	480000	m
LEAST	Raster Window	2410000	m
LNORTH	Raster Window	1910000	m
MEAST	Raster Window	2890000	m
MNORTH	Raster Window	2390000	m
SCALE	Map Scale	3000000	

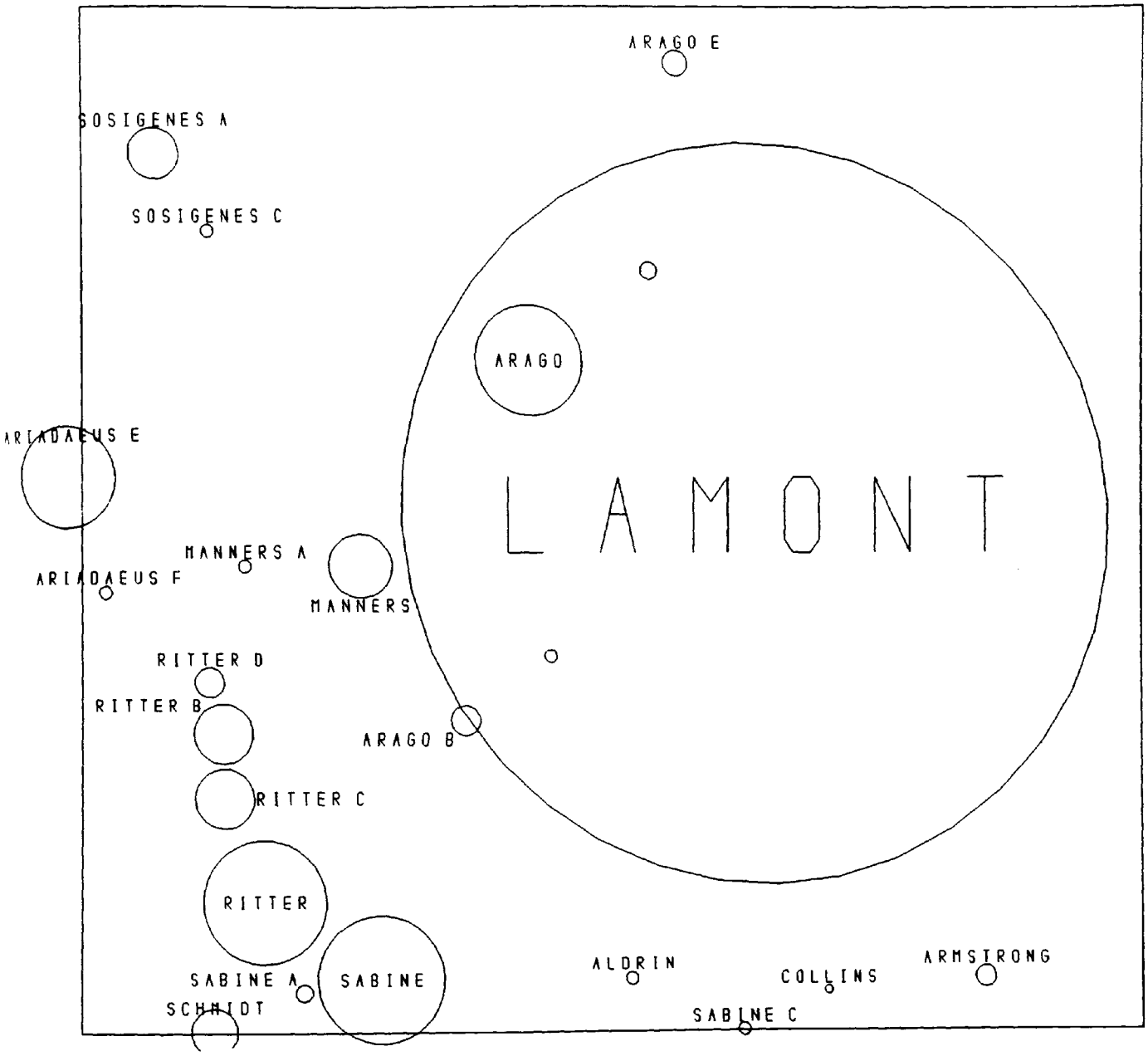
Fig 8.46 Moon's near side, scale 1:3,000,000.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	1980295	m
INORTH	Map Window	2480295	m
JEAST	Map Window	2319706	m
JNORTH	Map Window	2819706	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	339411	m
LEAST	Raster Window	1980295	m
LNORTH	Raster Window	2480295	m
MEAST	Raster Window	2319706	m
MNORTH	Raster Window	2819706	m
SCALE	Map Scale	2121320	

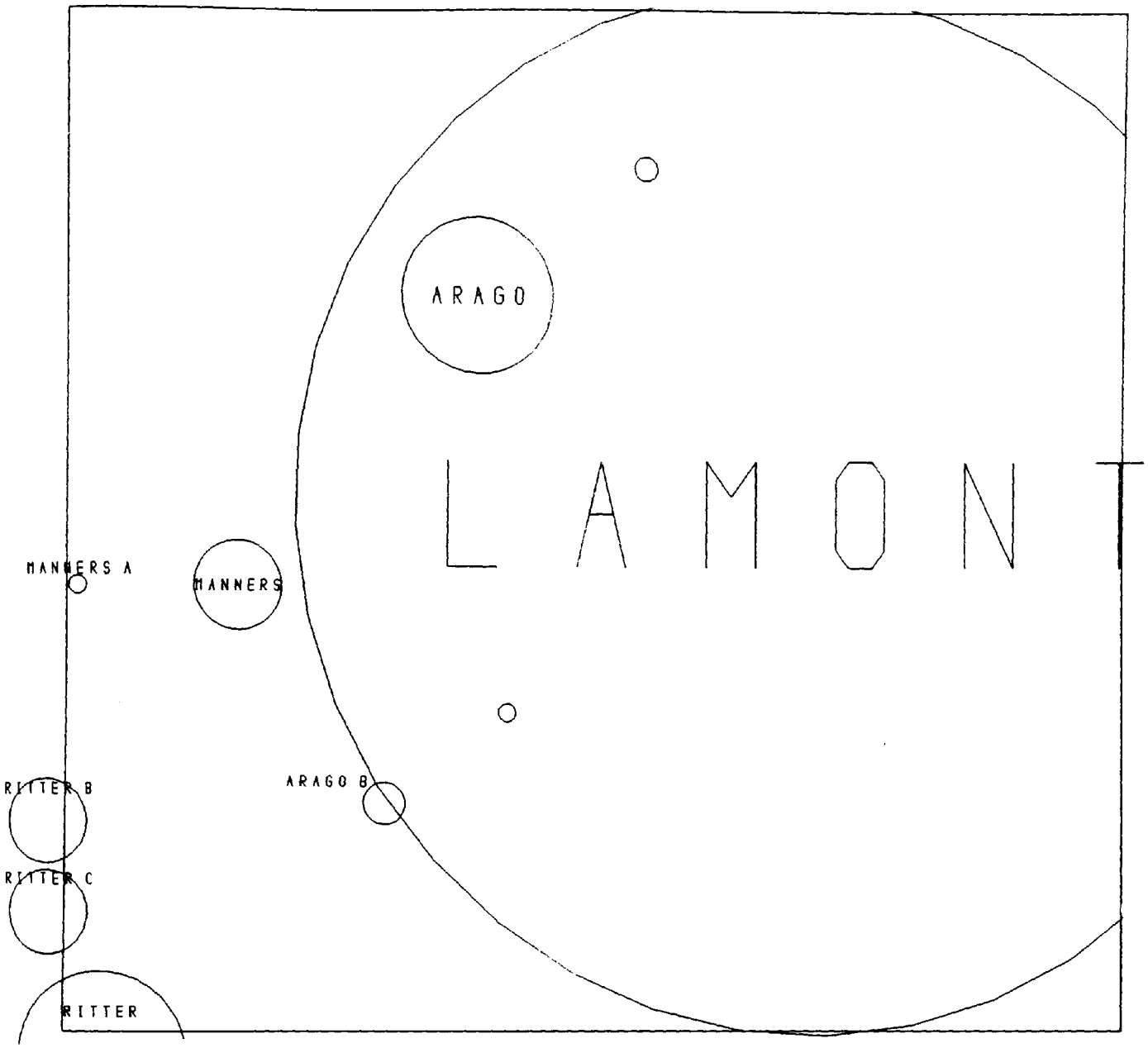
Fig 8.47 Moon's near side, scale 1:2,121,320.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	2530000	m
INORTH	Map Window	2030000	m
JEAST	Map Window	2770000	m
JNORTH	Map Window	2270000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	240000	m
LEAST	Raster Window	2530000	m
LNORTH	Raster Window	2030000	m
MEAST	Raster Window	2770000	m
MNORTH	Raster Window	2270000	m
SCALE	Map Scale	1500000	

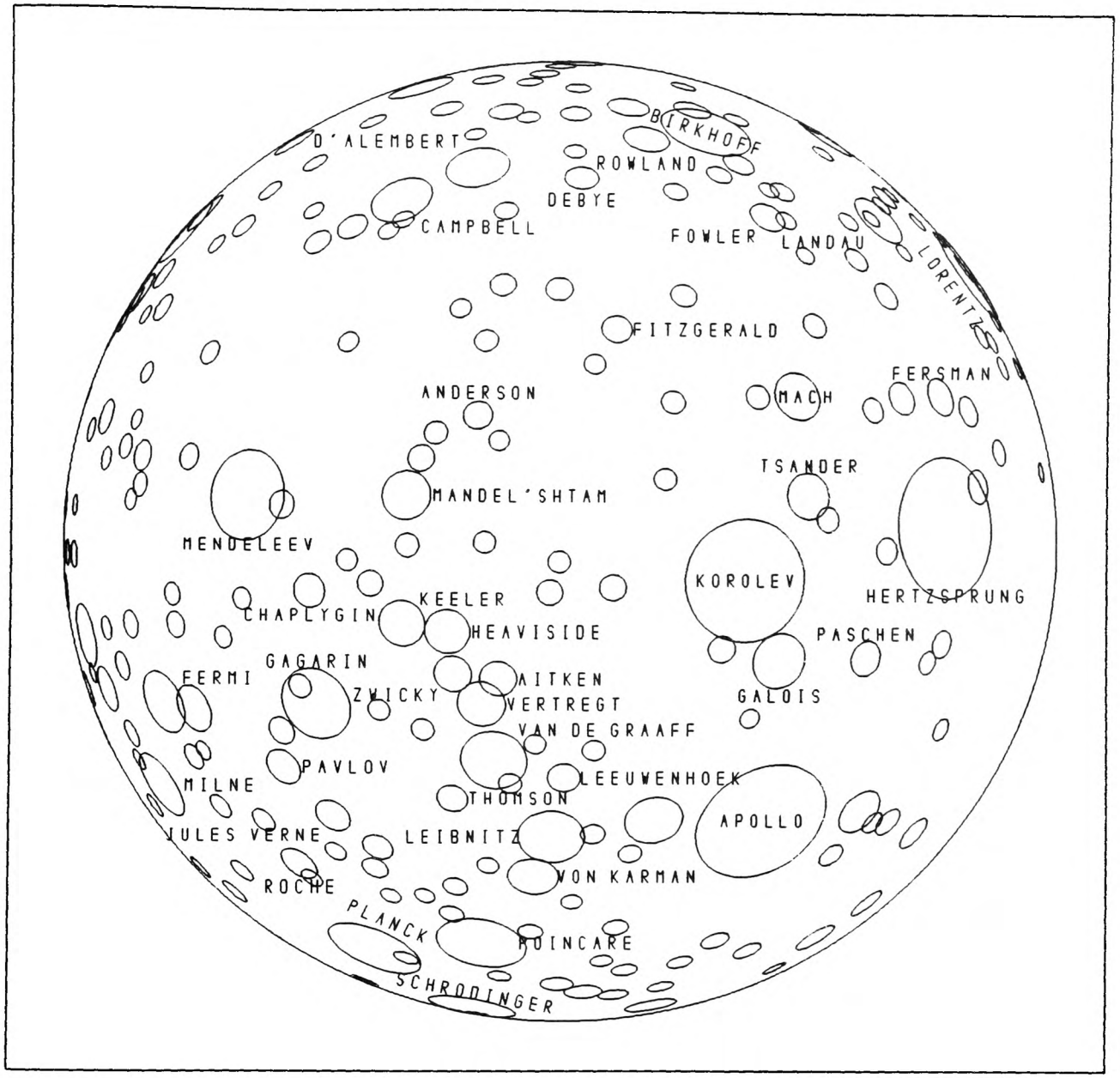
Fig 8.48 Moon's near side, scale 1:1,500,000.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	2565147	m
INORTH	Map Window	2065147	m
JEAST	Map Window	2734853	m
JNORTH	Map Window	2234853	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	169706	m
LEAST	Raster Window	2565147	m
LNORTH	Raster Window	2065147	m
MEAST	Raster Window	2734853	m
MNORTH	Raster Window	2234853	m
SCALE	Map Scale	1060660	

Fig 8.49 Moon's near side, scale 1:1,060,660.



Window Definition File "WIND_DEF" contents

RECORD NAME	DESCRIPTION	VALUE	UNITS
IEAST	Map Window	80000	m
INORTH	Map Window	80000	m
JEAST	Map Window	3920000	m
JNORTH	Map Window	3920000	m
IUNIT	Raster Size	500	pixels
ISIZE	Grid Sq. Size	3840000	m
LEAST	Raster Window	80000	m
LNORTH	Raster Window	80000	m
MEAST	Raster Window	3920000	m
MNORTH	Raster Window	3920000	m
SCALE	Map Scale	24000000	

Fig 8.50 Moon's far side, scale 1:24,000,000.

8.6 CHAPTER SUMMARY AND DISCUSSION

8.6.1 SUMMARY

This chapter has shown how a name placement system can be implemented through the use of high level rules in a logic program. This is achieved with the use of primitives for name placement and database handling. The three example maps which were tackled in this chapter show the versatility of the NAMEX system and PROLOG for solving the name placement problem on different types of maps and with different name placement rules.

An extensive range of high level rules have been implemented. The most notable of these include:

1) Road label selection based upon feature class, road length, and the junction importance.

2) Selecting road label configuration according to feature class, the approximate angle of the road, and the surrounding feature density.

3) Selecting point labels according to feature class and their proximity to important settlements.

4) The use of a variable radius of proximity in the case of settlement labels in feature dense areas.

5) The ability to increase settlement label size near to the coast.

6) The avoidance of placing settlement labels on the wrong side of a county boundary.

7) Selecting area label configuration according to the angle and elongation of the area.

8) Area label selection according to area size.

9) The avoidance of label ambiguity.

10) A new set of label positions and configurations for lunar craters.

11) The implementation of a form of generalisation.

12) Label abbreviation.

The ability to label different types of maps essentially depends upon label selection, label configuration selection, and label positioning and overlap detection rules. These can dramatically affect the appearance of the map as has been demonstrated in the three map examples covered. The name placement strategy

used is only a means of achieving the placement given the necessary set of rules such as those described in this chapter.

The strategy used, developed by the author, was selected for all three examples because it seemed a very capable means of solving each of the example name placement problems. The strategy is unique in that it allows the order of placement of labels to change dynamically in such a way that the label to be placed next is always the most difficult to place. However, it should be emphasized that users can use any name placement strategy of their choice.

8.6.2 IMPROVEMENTS

Regarding improvements to the NAMEX system, there are several label configurations which still need to be implemented. Also the help option does not function since it was not thought necessary for the purposes of research. However it can easily be amended to describe the options and available primitives.

One disadvantage found with the masking process is that when summing up pixel contents in a rectangular region in the raster image, if a feature has been masked out, any features not connected with this, but lying

underneath the mask, are also blanked out. Hence by masking out a road with a town at each end, the placement of the road label may not have any knowledge of the existence of the towns. Therefore it is always a good idea to make important features such as towns slightly larger in the raster image, so that when features such as roads are masked out, at least some pixels of the towns remain visible.

In the name placement strategy used for the three map examples, it might be possible to include other heuristics. For instance labels with equal numbers of positions and overlaps could be placed in order of the biggest labels first following along the lines of rule [2.73]. Also extra rules may need to be built in to avoid the unintelligent deletion of important feature labels in favour of the removal of less important labels.

One improvement which could be made to the Moon map would be to allow for secondary crater abbreviation to point towards their primary craters, to aid identification. Also it might be possible to include lines of latitude and longitude and investigate avoiding placing labels over these.

8.6.3 DEVELOPMENT AND RUN TIMES

The development time of a name placement system using NAMEX was only a few weeks as compared to approximately a year for the FORTRAN LABPOS program. On the other hand NAMEX is typically an order of magnitude or more slower than LABPOS (but still faster than manual placement). This slowness is a consequence of the use of a large number of rules in PROLOG and extensive backtracking. In fact because of the memory usage in the particular strategy used in PROLOG, the number of labels was limited to less than 200. Because of this, at present the role of NAMEX is best suited to the prototyping of name placement systems which can then be translated into much faster FORTRAN programs. However with faster hardware and more efficient logic programming languages becoming available, (Chapter 9), rule-based systems using the NAMEX system directly may become practical in the near future.

CHAPTER 9

CONCLUSION TO THE THESIS

9.1 INTRODUCTION

This thesis has described the work undertaken by the author in investigating automated name placement using rule-based systems. Two name placement systems were developed. The first, LABPOS, was an early prototype name placement system written in FORTRAN which was used to investigate both existing and new placement techniques. With the experience gained, NAMEX, a hybrid name placement system was implemented using PROLOG for high level rules and a FORTRAN program to access the low level cartographic data and name placement primitives. The NAMEX system provides an environment for the implementation of rule-based systems consisting of high level name placement rules.

To provided a suitable set of rules for the name placement systems, and to understand what information is needed to apply them, a study of manual name placement was undertaken and from this some previously unpublished name placement rules were extracted. Also a review of known automated name placement methods was carried out so as to compare different techniques, to select those which

were suitable for incorporation into the author's name placement systems, and to identify new areas for research.

In order for an automated name placement system to be able to produce maps of a comparable standard to a human cartographer, there are three requirements:

1) A rule-base of name placement rules.

2) Access to a similar level of cartographic data available to human cartographers.

3) A strategy for applying these rules.

9.1.1 RULES

There are such a large number, complexity and variety of name placement rules (only a very small selection of which were described in chapter 2) that if an automated system is going to be capable of placing names on a variety of maps it must be rule-based. This implies that its structure must allow for the editing of and inclusion of new name placement rules.

The early prototype system, LABPOS, included a user interface for the editing of a limited, pre-defined set of rules and specifications to control name placement.

However, the NAMEX system allows a greater degree of flexibilty in specifying relatively complex rules using logic programming. Several different types of rules, applied to the three map examples, were implemented using the NAMEX system. The rules apply to point, line and area labels and a mixture of all three. A small sample of these are described below.

In the Route Planner map example rules were used to perform spatial search. This was used to control label density in urban areas, label size in the vicinity of coastline, and the relationship between the proximity of a label to its feature and local map feature density.

In the same map example, rules were used in the selection of which line features to label according to their length, feature class, and the importance of junctions. Two different label configurations were available: horizontal and diagonal. These configurations were chosen according to criteria such as their proximity to important features, approximate line angle, and the class of the line.

The choice of area label configurations, horizontal or diagonal, was governed by rules which took into account the elongation and angle of the area concerned. This was successfully demonstrated on the Administrative Area map (Section 8.4). The selection of which areas to label was

accomplished by identifying areas big enough to take the label in at least one position.

Some of the rules took into account placement involving a mixture of label types and feature types. For instance it was possible to devise a rule which would help to decide whether a point label was on the same side of a county boundary as the point feature. Other rules governed how close different types of label could be placed to each other. In the Moon map example, the label positioning rules for craters allowed for the features to be treated as areas if the craters were large enough to contain them, otherwise as point features. Also in the Moon map example a generalisation rule was used to help decide which craters to plot at different scales.

9.1.2 CARTOGRAPHIC DATA

To implement these rules, cartographic data which the rules require must be freely available and preferably without large access time and memory storage overheads. If this is not the case, automated name placement systems cannot function to full rule capacity and satisfy map aesthetics. The NAMEX database structure, which used a combined vector and raster structure, was designed with these goals in mind. Database primitives were provided to extract the necessary high level information for these

rules.

9.1.3 STRATEGIES FOR NAME PLACEMENT

Several different name placement strategies were described in chapter 4. Most of these produced acceptable results for the maps they were designed for, although the aesthetic appeal of the placements varied in quality. However, different strategies are suited to different maps and although one which is designed for high density name placement would probably function on a low density map, it would do so less efficiently than one designed specifically for the low density map. In view of this, the ability to apply any specific strategy would be a welcome feature of an automated name placement system. The NAMEX system was devised so that this could be accomplished.

The name placement strategy used in LABPOS is iterative and makes use of a separation distance between labels to try and force them to be placed apart, but which can be relaxed under difficult circumstances. Weighting is applied to labels which are regularly in overlap so that these are placed over labels which have a greater freedom of movement to move out of overlap. After a label has undergone a user specified number of attempts at placement, such labels are fixed permanently in their most preferred position with respect to underlying detail, and

neighbouring labels in overlap which are still free to move are forced to find other positions.

The NAMEX system was used to demonstrate a new name placement strategy. The strategy concentrated on placing the most difficult labels first and, as it did so, took into account that the definition of the most difficult labels dynamically changed as each label was placed.

9.2 COMPARISON OF NAME PLACEMENT ON THE ROUTE PLANNER MAP USING LABPOS AND NAMEX

The use of PROLOG for the NAMEX system has several advantages over the FORTRAN prototype name placement system LABPOS. It is much more easily adaptable to a wide range of maps and the rules are easily changeable. The development time is much faster but a few weeks are still usually needed.

Two different name placement strategies were investigated, and these achieved different success rates. In LABPOS typically 80% of labels were placed unambiguously and 90% were placed without being in overlap with each other (Section 6.10). The name placement strategy used in NAMEX could (depending upon the rules) place nearly all labels unambiguously and without overlap,

but to achieve this it was allowed to delete labels as a last resort. It was found that the greater the number of rules incorporated into the NAMEX rule-based system, the slower the name placement process became. This is especially true when the label position definitions are encoded as PROLOG predicates instead of using the appropriate lower level FORTRAN called from the PROLOG primitives (Section 8.5.8). However this is still faster than manual placement rates.

At the time of writing, the NAMEX system would appear to be best suited for prototyping name placement systems designed for specific maps, but this is likely to change in the future with the introduction of more powerful hardware. If NAMEX is to be used for prototyping then once the rules for a prototype system have been implemented the rule-based system may be translated into a much faster FORTRAN program. Although the FORTRAN program could use existing primitives available in DB_ACCESS (Chapter 7), it would have to be capable of backtracking.

9.3 FURTHER IMPROVEMENTS TO NAMEX

9.3.1 NEW NAMEX PRIMITIVES

Several of the proposed primitives for the NAMEX system (the essential ones) were adequately demonstrated on three widely different types of maps. However some of

the proposed primitives remain to be implemented in future versions of NAMEX. In addition, during the author's construction of rule-based name placement systems using NAMEX, it became apparent that further improvements could be made to the NAMEX raster image primitives.

A possible new raster primitive would involve a means of determining the proximity of a feature using a "pixel probe". This could be extended from a location on the map, in any direction, until it hits the side of the feature concerned. The distance the probe has been extended would be returned, as the distance to the feature in metres.

Another useful raster primitive would be to find the biggest rectangle that would be able to fit into a particular region in the raster image. This would provide a short cut in the finding of available positions in which to place labels.

9.3.2 NEW STRATEGIES

One of the disadvantages of the NAMEX strategy used (Section 8.1.2 and 8.2.2) was that it was limited to less than 200 labels, otherwise a stack overflow occurred due to excessive backtracking. It may be possible to avoid this by the use of controlled backtracking so that a backtrack immediately goes to labels which are in

potential overlap with the current label which caused the backtrack. To do this, knowledge would be needed of when backtracking is taking place. This could be determined using a PROLOG fact which is asserted when a label cannot be placed (backtracking about to take place) and removed when a label can be placed (end of backtracking).

In the current NAMEX strategy the most recently placed label is deleted when the name placement process cannot progress any further. This is sometimes not the label which is really causing the problem. To help identify which label is the actual cause of the problem it might be possible to look for a recurring pattern in the labels during attempts at placement. Alternatively, perhaps candidate labels for deletion should be examined in the reverse order in which placement occurred until a label is found which is below a user specified threshold of importance.

9.4 ALTERNATIVE APPROACHES TO NAME PLACEMENT

9.4.1 EXPERT SYSTEMS

The task of extracting cartographic name placement expertise is formidable. Many of the rules used by cartographers are used subconsciously and so are very difficult to track down. This is particularly true for the

aesthetic appeal of name placement. It was for this reason that a name placement expert system was not implemented in the author's research.

One approach for discovering new name placement rules and for checking that cartographers' stated label position preferences are correct is to analyse maps statistically (Chapter 2). In fact NAMEX allows for the percentage occurrences for different configurations and positions to be included in the text definition rule-base.

9.4.2 MACHINE LEARNING APPROACH

A technique which may be applicable to name placement in the future is the machine learning approach. Unlike traditional expert systems which rely upon extracting knowledge from the experts, machine learning involves a system learning rules from its own experience (Cook, 1986).

One possible scenario would be for the name placement system to have a very simple set of name placement rules which are incorporated into a rule-based system such as NAMEX. This would then be used in a preliminary effort at name placement and afterwards the placed label data would be evaluated for quality of placement. The evaluation would consider the number of overlapping labels that

remain or have been deleted, the relative merits of label positions with respect to underlying and adjacent features, and the presentability of the map for instance avoiding aesthetic label placement errors such as point feature labels aligning with each other.

The results of the evaluation would be fed back and adjustments made to some parameters and placement rules, before repeating the name placement. The process would be repeated until either sufficient quality is attained, or no further significant improvement can be achieved. The rules and parameters adapted for the final result would be retained as knowledge for applications to future maps of that type.

Another approach to the use of machine learning for solving name placement problems might be to use "neural networks" which are a radically different approach to computing which mimics the way neurons are organised in the brain (Boothroyd, 1988). These have proven capable of learning to recognise spoken words and recognising faces.

9.4.3 PARALLEL PROCESSING

Although manual name placement is a sequential process which involves backtracking, there could be significant improvements in speed if an automated name

placement system was implemented on a parallel processing computer system such as a configured array of transputers (Dettmer, 1986). This could function by dividing the labels equally between the transputers, solving the name placement problem for each group of labels on a transputer and then checking between each transputer for overlaps between groups which would need to be corrected.

9.4.4 NATURAL LANGUAGE INTERFACE

On comparing the logic pseudo code in chapter 8 with the actual PROLOG programs, the reader will have realised that the writing of name placement programs using the NAMEX system still relies heavily upon knowledge of the database structure and upon technicalities of PROLOG programming. It may be possible, through the careful use of the available primitives, to develop some form of very high level natural name placement language. In order for a user to be oblivious to the presence of the database structure and logic programming efficiency, rules would have to be embedded into the NAMEX system to take these into account.

9.5 CONCLUDING REMARKS

The FORTRAN program, LABPOS, has been given to the Ordnance Survey and some trial name placement experiments undertaken. Although name placement on the Ordnance Survey Route Planner map is no longer required by the Ordnance Survey as they have already decided upon where to place the names, it appears that the program may be suitable for name placement on new maps using the same database structure (King, 1988).

One interesting question which arises about automated name placement systems is their useful role in cartographic organisations. Such systems were originally constructed to see if automated name placement on maps was practical although some of the earlier attempts were both map specific, primitive and required manual editing for unresolved name placement conflicts. In the years that have followed, automated name placement systems have improved to such an extent that labels can be deleted and aesthetics can be taken into account. However to the author's knowledge no name placement system has been built which can generate complicated map products which are indistinguishable from those produced manually by expert cartographers. This is because of the problems associated with extracting all the name placement knowledge possessed by skilled cartographers and also the fact that cartographic standards acceptable to cartographic

organisations are very high.

The main advantage of automated name placement systems is that they are much faster than their human counterparts. If cartographic organisations are prepared to lower their expectations of name placement slightly to a level which, although not thought highly of by cartographic experts, would satisfy the requirements of the layman, then automated name placement should be a viable proposition and lead to increased efficiency.

Another interesting aspect of automated name placement realised by the author was that it was not the placement strategy which had the most effect on the appearance of the map, but instead the selection of labels and configurations and positions. Future name placement systems should concentrate on this aspect of the problem if they are to improve upon existing systems.

Finally, the author believes that the potential of automated name placement systems has not been fully recognised. For instance, systems such as NAMEX should be capable of handling many other forms of graphical diagrams (Pfefferkorn et al) such as electronic circuits, computer graphics and technical drawings, all of which need to be labelled.

APPENDIX 1

Table 1 Frequency of occurrence of different point label positions on the Ordnance Survey Route Planner map.

Label Position	Library grid squares 201		301		501		208		Total	
1	17	(13%)	38	(16%)	33	(17%)	13	(14%)	101	(15%)
2	11	(8%)	11	(5%)	17	(9%)	4	(4%)	43	(6%)
3	1	(1%)	2	(1%)	5	(3%)	3	(3%)	11	(2%)
4	3	(2%)	9	(4%)	11	(6%)	7	(7%)	30	(4%)
5	3	(2%)	3	(1%)	6	(3%)	4	(4%)	16	(2%)
6	8	(6%)	17	(7%)	7	(4%)	2	(2%)	34	(5%)
7	3	(2%)	2	(1%)	0	(0%)	1	(1%)	6	(1%)
8	3	(2%)	7	(3%)	5	(3%)	2	(2%)	17	(3%)
9	0	(0%)	2	(1%)	6	(3%)	6	(6%)	14	(2%)
10	5	(4%)	12	(5%)	7	(4%)	4	(4%)	28	(4%)
11	11	(8%)	24	(10%)	18	(9%)	8	(8%)	61	(9%)
12	13	(10%)	15	(6%)	3	(2%)	4	(4%)	35	(5%)
13	5	(4%)	6	(2%)	3	(2%)	5	(5%)	19	(3%)
14	4	(3%)	6	(2%)	5	(3%)	3	(3%)	18	(3%)
15	2	(2%)	4	(2%)	3	(2%)	1	(1%)	10	(1%)
16	6	(5%)	17	(7%)	17	(9%)	1	(1%)	41	(6%)
17	6	(5%)	5	(2%)	3	(2%)	2	(2%)	16	(2%)
18	7	(5%)	9	(4%)	12	(6%)	11	(12%)	39	(6%)
19	3	(2%)	17	(7%)	10	(5%)	1	(1%)	31	(5%)
20	22	(17%)	37	(15%)	28	(14%)	13	(14%)	100	(15%)
Total	133		243		199		95		670	

(Percentage usage of a particular label position out of the total number of labels is given in brackets)

Table 2 Frequency of occurrence of configurations of A class road labels.

Label Config.	Library grid squares 201		301		501		208		Total	
1	3	(9%)	3	(3%)	7	(6%)	2	(6%)	15	(5%)
2	5	(14%)	2	(2%)	9	(7%)	3	(9%)	19	(6%)
3	3	(9%)	14	(13%)	23	(19%)	2	(6%)	42	(14%)
4	1	(3%)	14	(13%)	9	(7%)	1	(3%)	25	(8%)
5	3	(9%)	17	(16%)	12	(10%)	0	(0%)	32	(11%)
6	1	(3%)	3	(3%)	3	(2%)	0	(0%)	7	(2%)
7	4	(11%)	7	(7%)	11	(9%)	0	(0%)	22	(7%)
8	8	(23%)	26	(25%)	31	(25%)	22	(67%)	87	(29%)
9	3	(9%)	14	(13%)	6	(5%)	2	(6%)	25	(8%)
10	1	(3%)	2	(2%)	2	(2%)	0	(0%)	5	(2%)
11	3	(9%)	4	(4%)	11	(9%)	1	(3%)	19	(6%)
Total	35		106		124		33		298	

(Percentage usage of a particular label configuration out of the total number of labels is given in brackets)

Table 3 Frequency of occurrence of A class road label angles on the Ordnance Survey Route Planner map.

Angles (degrees)	Library grid squares 201		301		501		208		Total	
-90 :-70	1	(3%)	2	(2%)	7	(6%)	1	(3%)	11	(4%)
-70 :-50	1	(3%)	3	(3%)	7	(6%)	3	(9%)	14	(5%)
-50 :-30	0	(0%)	2	(2%)	2	(2%)	2	(6%)	6	(2%)
-30 :-10	3	(9%)	4	(4%)	10	(8%)	1	(3%)	18	(6%)
-10 : 10	25	(71%)	70	(66%)	85	(69%)	15	(45%)	195	(65%)
10 : 30	0	(0%)	8	(8%)	4	(3%)	2	(6%)	14	(5%)
30 : 50	0	(0%)	10	(9%)	4	(3%)	3	(9%)	17	(6%)
50 : 70	2	(6%)	3	(3%)	3	(2%)	4	(12%)	12	(4%)
70 : 90	3	(9%)	4	(4%)	2	(2%)	2	(6%)	11	(4%)
	35		106		124		33		298	

(Percentage usage of a particular label range of angles out of the total number of labels, is given in brackets)

Table 4 Frequency of occurrence of configurations of B class road labels.

Label Config.	Library grid squares 201	301	501	208	Total
1	6 (33%)	16 (41%)	18 (47%)	7 (41%)	47 (42%)
2	7 (39%)	17 (44%)	11 (29%)	5 (29%)	40 (36%)
3	2 (11%)	4 (10%)	4 (11%)	2 (12%)	12 (11%)
4	3 (17%)	2 (5%)	5 (13%)	3 (18%)	13 (12%)
Total	18	39	38	17	112

(Percentage usage of a particular label configuration out of the total number of labels is given in brackets)

Table 5 Frequency of occurrence of B class road label angles on the Ordnance Survey Route Planner map.

Angles (degrees)	Library grid squares 201	301	501	208	Total
-90 :-70	6 (33%)	4 (10%)	3 (8%)	1 (6%)	14 (13%)
-70 :-50	2 (11%)	7 (18%)	3 (8%)	1 (6%)	13 (12%)
-50 :-30	2 (11%)	6 (15%)	0 (0%)	0 (0%)	8 (7%)
-30 :-10	2 (11%)	0 (0%)	4 (11%)	1 (6%)	7 (6%)
-10 : 10	1 (6%)	6 (15%)	11 (29%)	1 (6%)	19 (17%)
10 : 30	3 (17%)	5 (13%)	2 (5%)	2 (12%)	12 (11%)
30 : 50	1 (6%)	2 (5%)	4 (11%)	4 (24%)	11 (10%)
50 : 70	0 (0%)	4 (10%)	6 (16%)	6 (35%)	16 (14%)
70 : 90	1 (6%)	5 (13%)	5 (13%)	1 (6%)	12 (11%)
	18	39	38	17	112

(Percentage usage of a particular label range of angles out of the total number of labels, is given in brackets)

Table 6 Frequency of occurrence of configurations of river labels.

Label Config.	Library grid squares 201	301	501	208	Total
1	5 (33%)	9 (32%)	5 (33%)	4 (21%)	23 (30%)
2	4 (39%)	12 (43%)	1 (7%)	7 (37%)	24 (31%)
3	4 (11%)	4 (14%)	5 (33%)	5 (26%)	18 (23%)
4	2 (17%)	3 (11%)	4 (27%)	3 (16%)	12 (16%)
Total	15	28	15	19	77

(Percentage usage of a particular label configuration out of the total number of labels is given in brackets)

Table 7 Frequency of occurrence of river label angles on the Ordnance Survey Route Planner map.

Angles (degrees)	Library grid squares 201	301	501	208	Total
-90 :-70	2 (13%)	2 (7%)	2 (13%)	0 (0%)	6 (8%)
-70 :-50	2 (13%)	5 (18%)	1 (7%)	0 (0%)	8 (10%)
-50 :-30	0 (0%)	4 (14%)	1 (7%)	1 (5%)	6 (8%)
-30 :-10	2 (13%)	5 (18%)	3 (20%)	4 (21%)	14 (18%)
-10 : 10	1 (7%)	0 (0%)	2 (13%)	2 (11%)	5 (6%)
10 : 30	1 (7%)	3 (11%)	1 (7%)	2 (11%)	7 (9%)
30 : 50	3 (20%)	1 (4%)	2 (13%)	5 (26%)	11 (14%)
50 : 70	2 (13%)	6 (21%)	2 (13%)	4 (21%)	14 (18%)
70 : 90	2 (13%)	2 (7%)	1 (7%)	1 (5%)	6 (8%)
	15	28	15	19	77

(Percentage usage of a particular label range of angles out of the total number of labels, is given in brackets)

Table 8 Administrative Area Label Statistics

European Constituency areas (England and Wales only)

Non-split:	38	
Split (centred):	2	(1 is diagonal)
Split (EC on lower line):	9	(1 1s diagonal)
Outside the map:	3	
Numbered labels:	16	

(All of these labels unless otherwise stated are horizontal)

County areas (Great Britain)

Horizontal (Non-split):	52	
Horizontal (Split):	6	
Diagonal:	1	
Curved:	9	

District areas (Great Britain)

Horizontal (Non-split):	308	(24 are arrowed)
Horizontal (Split):	40	(3 are arrowed)
Diagonal:	68	(8 are split)
Curved:	2	

APPENDIX 2

2.0 INTRODUCTION TO PROLOG

PROLOG is a logic programming language and for reasons given in the introduction has been selected for use in the author's research into automated name placement. The three basic statements in logic programs are: facts, rules and queries (Sterling and Shapiro). These will be introduced to the reader in this appendix using a cartographic example dealing with cities, roads and other attributes. A good introduction to PROLOG is "The Art of PROLOG", by Sterling and Shapiro.

2.1 FACTS

"A fact is the simplest statement in PROLOG" (Sterling and Shapiro). For instance, to state the fact that "cardiff" is by the sea, one could say:

```
seaside(cardiff).
```

"seaside" is the relationship or predicate. The predicate applies to the item or atom inside the curved brackets. Note that both predicates and atoms always start with small letters. A full stop is used to terminate a predicate.

Facts can contain more than one atom and atoms can include numbers. If we wanted to enter some facts about the locations of various cities in Great Britain, then these could be entered thus:

```
/* city, country, km east, km north */
location(cardiff,wales,317,177).
```

In the above example, the "location" predicate indicates that "cardiff" is in the country of "wales" and lies at coordinates 317km east and 177km north. Note that comment statements have been used. These are easily recognisable because they are sandwiched between "/*" and "*/" statements. Commas are used to separate the atoms inside the predicate. Commas are also used to separate variables and lists inside predicates (see below).

Lists of atoms can also be included in facts and are easily identified since they lie between square brackets. For example a fact giving all the main roads leading out of cardiff could be expressed thus:

```
main_roads(cardiff,[m4,a469,a48,a4160]).
```

The very first element of the list, "m4" in the above

case, is known as the head of the list, the remainder of the list "a469,a48,a4160", is known as the tail. Lists can also contain lists and these can contain other lists and so on. For instance the predicate "facilities" contains data on the main facilities at different places:

```
facilities([[cardiff,[airport,university,castle,port]],
  [aberystwyth,[university]],
  [swansea,[university,castle,port,airport]],
  [plymouth,[polytechnic,port,airport]],
  [london,[castle,university,polytechnic,port,airport]],
  [edinburgh,[university,castle,airport]],
  [york,[university]],
  [dover,[port,castle]],
  [bristol,[port,university,airport]],
  [birmingham,[university,polytechnic,airport]]]).
```

2.2 QUERIES

It is possible to query facts and indeed rules when presented with the PROLOG prompt "?- ". For instance:

```
?- seaside(cardiff).
```

is true and this is indicated by a "yes" appearing after pressing the carriage return. If this were not true, or the information was unavailable, then a "no" would be returned.

Variables can also be included in queries. A variable can be recognized because its first letter is a capital. To demonstrate the use of variables the following facts are included about 6 places:

```
seaside(cardiff).
seaside(aberystwyth).
seaside(swansea).
seaside(plymouth).
seaside(dover).
seaside(bristol).
```

To find out all the places at the seaside one can type the following and keep on hitting ";" until no more answers are available.

```
?- seaside(X).
X = cardiff ?;
X = aberystwyth ?;
X = swansea ?;
X = plymouth ?;
X = dover ?;
X = bristol ?;
```

Just hitting the carriage return key without a ";" will only return one answer. The ";" causes backtracking to occur if alternative answers are available.

The heads and tails of lists can be extracted using variables, for instance:

```
?- main_roads(_,[H|T]).
H = m4
T = [a469,a48,a4160]
```

To do this the "|" or pipe character is used to separate the head from a tail in the list. Also note in the example above the use of "_". This is used when we are not interested in the contents of a particular atom or list in a relationship.

2.3 RULES

Rules are constructed using the ":-" or logical implication and logical "," (AND) and ";" (OR) operators. A simple example would be the definition of whether one place is east of another:

```
east_of(Place1,Place2):-
   location(Place1,_,East1,_),
   location(Place2,_,East2,_),
   East1 > East2.
```

This means Place1 is "east_of" Place2 IF the location of Place1 is available AND the location of Place2 is available AND the easting of Place1 is greater than the easting of Place2. "west_of", "north_of" and "south_of" can be defined similarly.

Allowing for some additional facts:

```
location(swansea,wales,264,193).
location(london,england,530,180).
```

Rules can be used in queries thus:

```
?- east_of(cardiff,swansea).
yes

?- east_of(cardiff,london).
no

?- east_of(cardiff,X).
X = swansea
```

Rules can also be made up out of other rules as is illustrated below with the "in_between" predicate:

"Place X is in between Place Y and Place Z"

```
in_between(X,Y,Z):-
   location(X,_,Eastx,_),
   location(Y,_,Easty,_),
   location(Z,_,Eastz,_),
   (east_of(X,Y),west_of(X,Z));
    (east_of(X,Z),west_of(X,Y)).
```

Note that when using the ";" (OR) operand, brackets are used to enclose the expressions involved.

If predicates do not succeed, PROLOG has the ability to backtrack automatically in order to see if alternative answers exist. If this facility is not required, the "!" or cut symbol is placed in the predicate concerned and effectively limits the predicate to one answer. However, sometimes a user may wish the predicate to determine all the answers available. In interactive mode this is achieved by pressing the ";" key. The same effect can be achieved in a rule by placing a "fail" on the end of the predicate concerned.

Rules can even involve recursion, and this is particularly useful in the processing of lists where elements within the list need to be extracted. The "member(Element,List)" predicate is used for this purpose and is defined thus:

```
member(Element,[Element|Tail]).
member(Element,[Head|Tail]):-
   member(Element,Tail).
```

The way it works is to recursively call itself to see if "Element" is contained within the "List". The list concerned is gradually decomposed by pulling off the head and re-submitting the tail until either the Element is found at the head of the list (top line), or no elements remain in the list. If the latter occurs, then the predicate fails thus indicating that Element does not exist in List. If the former occurs then the predicate succeeds.

2.4 OTHER USEFUL PREDICATES

This section of the appendix gives a brief summary of some commonly used predicates in this thesis. To demonstrate these the facts and rules at the bottom of appendix 2 are used:

append(List1,List2,Out_list).

This appends two input lists together to produce an output list. e.g. list all the roads connected to cardiff and aberystwyth:

```
?- main_roads(cardiff,List1),
   main_roads(aberystwyth,List2),
   append(List1,List2,Out).

List1 = [m4,a469,a48,a4160]
List2 = [a487,a44]
Out   = [m4,a469,a48,a4160,a487,a44]
```

pull_off(Element,List,N).

Pulls off the first N elements of a List during backtracking. e.g. find the places where the m4 is in the first three main roads in each list:

```
?- main_roads(Place,List),
   pull_off(m4,List,3).

Place = cardiff;
Place = swansea;
Place = bristol;
```

reverse(List1,List2).

reverses List1 and places it in List2. e.g.

```
?- main_roads(cardiff,List),
   reverse(List,Rev_list).

List = [m4,a469,a48,a4160]
Rev_list = [a4160,a48,a469,m4]
```

select(Element,List,Remainder).

This is very much like "member", except that a remainder list is returned. Also, unlike "member", it can be used safely in a recursive predicate. e.g. select all roads except the a48.

```
?- main_roads(cardiff,List),
   select(a48,List,Remainder).

List = [m4,a469,a48,a4160]
Remainder = [m4,a469,a4160]
```

select_first(Element,List,Remainder).

is like select but only returns one element from the list and does not backtrack.

sort(List,Slist).

This is used to sort the elements of a list alphabetically/numerically. e.g.

```
?- main_roads(cardiff,List),
   sort(List,Slist).

List = [m4,a469,a48,a4160]
Slist = [a4160,a469,a48,m4]
```

convert(No1,Units1,No2).

This can be used to convert a distance in specified units into metres.

```
?- convert(10,miles,X).

X = 16090
```

loop(Count,Start,Finish).

This predicate will repeatedly loop from Start to Finish, when being forced to backtrack, in increments of one which are counted by Count. e.g. convert numbers between 1 and 5 miles into metres:

```
?- loop(N,1,5),
   convert(N,miles,X),
   write(X),nl,
   fail.

1609
3218
4827
6436
8045
```

Note that "fail" has been used to force "loop" to backtrack and the results of the conversion have been output to the screen with "write". "nl" is used to print a new line on the screen.

2.5 PROGRAM EXAMPLE

```
seaside(cardiff).
seaside(aberystwyth).
seaside(swansea).
seaside(plymouth).
seaside(dover).


location(cardiff,wales,317,177).
location(aberystwyth,wales,258,282).
location(swansea,wales,264,193).
location(plymouth,england,247,57).
location(london,england,530,180).
location(edinburgh,scotland,324,674).
location(york,england,460,453).
location(dover,england,631,142).
location(bristol,england,358,173).
location(birmingham,england,407,287).


main_roads(cardiff,[m4,a469,a48,a4160]).
main_roads(aberystwyth,[a487,a44]).
main_roads(swansea,[m4,a4067,a4070,a483]).
main_roads(plymouth,[a38,a388,a386]).
main_roads(london,[a1,a2,m20,m25,m3,m4,m40,m10,m11,a12,a41
,a4]).
main_roads(edinburgh,[a1,m8,a8,a71,a89,a71,a702]).
main_roads(york,[a19,a59,a64,a166,a1079]).
main_roads(dover,[a2,a20,a258,a256]).
main_roads(bristol,[m4,m5,a420,a4,a37,a38,a370]).
main_roads(birmingham,[m5,m54,m6,a38,a456,a435,a45,a41]).


east_of(Place1,Place2):-
   location(Place1,_,East1,_),
   location(Place2,_,East2,_),
   East1 > East2.

west_of(Place1,Place2):-
   location(Place1,_,East1,_),
   location(Place2,_,East2,_),
   East1 < East2.

north_of(Place1,Place2):-
   location(Place1,_,_,North1),
   location(Place2,_,_,North2),
   North1 > North2.

south_of(Place1,Place2):-
   location(Place1,_,_,North1),
   location(Place2,_,_,North2),
   North1 < North2.
```

APPENDIX 3

3.0 NAME PLACEMENT PRIMITIVES

This appendix gives brief descriptions of the primitives used in the NAMEX system. Sometimes two primitives exist with the same name. The one with the least number of parameters uses the current FORTRAN variable contents and so does not have to read a new record.

Four commonly used variables will be: "Fsn", "Ftype", X, and Y. These correspond to feature serial number (cartographic feature record), feature type (0, 1, 2 or point, line, area) and X and Y coordinates of a label/feature/location on the map (metres or pixels depending upon the nature of the primitive).

The primitives fall into seven classes:

1) NAMEX menu control.

2) Initialisation of the NAMEX system.

3) Database access.

4) Name placement.

5) Validation.

6) Text processing.

7) Useful PROLOG predicates.

3.1 NAMEX MENU CONTROL

heading. - Prints out the NAMEX user menu.

do(1). - Menu option 1, loads up cartographic data.

do(2). - Selects which labels to place.

do(3). - Carries out name placement.

do(4). - Saves name placement data.

do(5). - Edits name configurations rules/parameters.

do(6). - Help facility.

do(7). - Exit from NAMEX.

3.2 INITIALISATION OF THE NAMEX SYSTEM.

title. - This activates the initialisation of the NAMEX system and calls the menu.

set_up. - This loads up POPLOG libraries and sets up the link between PROLOG and FORTRAN.

load_vector_data. - Loads cartographic vector data.

load_raster_image. - Loads raster image data.

3.3 DATABASE ACCESS.

3.3.1 NAME/LABEL DATA.

close_label_output. - Closes the "LABEL_OUTPUT" file.

get_name_asc_value(N,Let_pos,Asc) &
get_name_asc_value(Let_pos,Asc). - Gets the ASC-II value of the character at position Let_pos in the name (record N) in the "NAME" file.

get_name_def_fcode(N,Fcode) &
get_name_def_fcode(Fcode). - Gets the feature code of record (N) in the "NAME_DEF" file.

get_name_def_fsn(N,Fsn) &
get_name_def_fsn(Fsn). - Gets the feature serial number of record (N) in the "NAME_DEF" file.

get_name_def_ftype(N,Fcode) &
get_name_def_ftype(Fcode). - Gets the feature type (0, 1, 2) of a record (N) in the "NAME_DEF" file.

get_name_def_name_pointer(N,Npoint) &
get_name_def_name_pointer(Npoint). - Gets the pointer to the appropriate record in the "NAME" file for a record (N) in the "NAME_DEF" file.

get_name_letter_length(N,Let_count) &
get_name_letter_length(Let_count). - Gets the number of characters for a name (record N) in the "NAME" file.

get_name_name_def_pointer(N,Npoint). &
get_name_name_def_pointer(Npoint). - Gets the "NAME" file (record N) pointer to the "NAME_DEF" ("MULTI_FEAT") file.

get_name_no_of_occurrences_of(N,No). &
get_name_no_of_occurrences_of(No). - Gets the number of versions of the same name (record N) in the "NAME" file.

get_names(Npoint,No,Pointer). - Given the "NAME" pointer (Npoint) to the "NAME_DEF" ("MULTI_FEAT") file and the number (No) of features with the same name, this primitive

finds (upon backtracking) the appropriate pointers to the "NAME_DEF" file.

get_text_param_font(Fcode,F). - Gets the recommended font number (F) for the specified feature code from the parameterized text definition rule-base.

grab_name(N,Text). &
grab_name(Text). - Gets the name (text version) for the "NAME" file record (N).

increment_label_record. - Increments (and writes) "LABEL" file record.

initialise_label_counter. - **Sets label record counter to** 1.

label_removed(L). - Checks to see if a label (L) has been "removed" from the "LABEL" file.

load_text_param. - Loads parameterized text definition rule-base.

open_label_output. - Opens the "LABEL_OUTPUT" file.

output_label(N). - Outputs label N to the "LABEL_OUTPUT" file.

output_label_count(N). - Total number (N) of records output to the "LABEL_OUTPUT" file.

read_label_details(Param_list,Out_list). &
read_label_details(L,Param_list,Out_list). - Enter the field numbers of required "LABEL" (record L) details in a list (Param_list) and these will be returned in the list (Out_list).

read_text_param(Fcode,Param1,Param2,Value_out). - Reads parameterized text definition rule-base single field contents using the feature code and parameters (Param1 and Param2) to index the requested data.

read_text_param(Fcode,Param_list,Out_list). - Enter the field numbers of the required parameterized text definition rule-base details in a list (Param_list), for a particular feature code, and these will be returned in the list (Out_list).

remove_label(L). - Removes a label (L) from the "LABEL" file.

save_text_param. - Saves parameterized text definition rule-base.

write_label_details(Param_list,In_list). &
write_label_details(Param_list,In_list). - Enter the field

numbers of required "LABEL" (record L) details in a list (Param_list), and the contents held in the list (In_list) will be written to the label record.

write_text_param(Fcode,Param1,Param2,V). - Writes a value (V) to the parameterized text definition rule-base, using the feature code and parameters (Param1 and Param2) as an index.

3.3.2 VECTOR DATA

3.3.1 GENERAL

get_feature_code(Ftype,Fsn,Fcode) &
get_feature_code(Ftype,Fcode). - Gets the feature code for a given feature serial number and feature type.

get_feature_name_def_pointer(Ftype,Fsn,Npoint) &
get_feature_name_def_pointer(Ftype,Npoint). - Gets the pointer to a record in the "NAME_DEF" file given the feature serial number and the feature type.

3.3.2.2 POINTS

get_point_coords(Fsn,East,North) &
get_point_coords(East,North). - Gets "POINT" coordinates.

get_point_line_pointer(Fsn,Lpoint) &
get_point_line_pointer(Lpoint). - Gets the pointer to the "LINK_NODE" file.

get_point_no_of_lines(Fsn,No) &
get_point_no_of_lines(No). - Gets the number of lines (No) joining at the point.

read_point_details(Param_list,Var_list). &
read_point_details(Fsn,Param_list,Var_list). - Enter the field numbers of required "POINT_DEF" details in a list (Param_list) and these will be returned in the variable list (Var_list).

3.3.2.3 LINES

get_line_coord_list(Fsn,List). - Gets the coordinates of a line and returns these in a List of sub-lists of coordinate pairs.

get_line_fsn(N,Fsn). - Gets the "LINE_DEF" feature serial number from the appropriate record (N) in the "LINK_NODE" file.

get_line_length(Fsn,Length) &
get_line_length(Length). - Gets the length of the line.

get_line_nodes(Fsn,Node1,Node2) &
get_line_nodes(Node1,Node2). - Gets the pointers to the nodes in the "POINT_DEF" file.

read_line_details(Param_list,Var_list) &
read_line_details(Fsn,Param_list,Var_list). - Enter the field numbers of required "LINE_DEF" details in a list (Param_list) and these will be returned in the variable list (Var_list).

3.3.2.4 AREAS

get_area_area(Fsn,Area) &
get_area_area(Area). - Get area of area.

get_area_cm_(Fsn,East,North) &
get_area_cm_(East,North). - Gets coordinates of area centre of mass.

get_area_elong(Fsn,Elong) &
get_area_elong(North). - Get elongation of area.

get_area_orient(Fsn,Ang) &
get_area_orient(Ang). - Get orientation of area (degrees).

get_area_limits(Fsn,Min_east,Max_east,Min_north,Max_north) &
get_area_limits(Min_east,Max_east,Min_north,Max_north).
- Gets areas minimum and maximum eastings and northings.

get_area_rast(N,East,North,Length) &
get_area_rast(East,North,Length). - Gets the easting, northing and length of an area run-length strip (N).

get_area_seed_(Fsn,East,North) &
get_area_seed_(East,North). - Gets coordinates of area seed.

read_area_details(Param_list,Var_list). &
read_area_details(Fsn,Param_list,Var_list). - Enter the field numbers of required "AREA_DEF" details in a list (Param_list) and these will be returned in the variable list (Var_list).

3.3.3 RASTER DATA.

erase_mask. - Remove current mask.

mask_out_current_feature. - Masks out the current feature.

mask_out_feature(Fsn,Ftype). - Mask out feature (Fsn = feature serial number, Ftype = feature type (0, 1, 2).

rast_circle_init(X,Y,R). - Examines a circular region of

radius R pixels in the raster image.

raster_sum(N,Sum). - Sums up the priority of pixels, in bit planes 1 to N, of features contained within the rasterized region under examination.

rast_free_pix(N). - Total number of free space pixels in the rasterized region under examination.

rast_over_edge(N). - Total number of pixels beyond map edge in the rasterized region under examination.

rast_plane_read(P,Value). - Read the pixel contents of a specified raster plane (P) in the rasterized region under examination.

rast_point_init(X,Y). - Examines a pixel location in the raster image.

rast_rect_init(X,Y,L,W,Ang). - Examines a rectangular region of length L (pixels), width W (pixels) and of angle Ang (degrees), in the raster image.

window_pixel_area(Pixel_sum). - Total number of pixels within a rasterized region under examination.

window_pixel_sum(Pixel_sum). - Similar to "raster_sum", but looks at all bit planes in use.

3.4 NAME PLACEMENT.

compute_current_label_dimensions. - Computes the dimensions of the current label, using current attribute values.

compute_label_dimensions(N,F,Ltsep,Wdsep,Lnsep,Labln). - Computes the dimensions of a label (length = Labln) using the "NAME" file record (N) for the label, the font number (F), letter separation (Ltsep), the word separation (Wdsep) and the label line separation (Lnsep).

compute_label_dimensions(F,Ltsep,Wdsep,Lnsep,Labln). - Computes the dimensions of the current label (length = Labln) given the font number (F), letter separation (Ltsep), the word separation (Wdsep) and the label line separation (Lnsep).

compute_label_position(Fsn,Ftype,Pos,Prox,Config). - Computes label positions in FORTRAN. Requires the feature serial number, feature type (0,1,2), label position index number (1-20), radius of proximity in metres (0 if not applicable) and the label configuration number. N.B. The dimensions of a label should always be determined and the appropriate label record read before computing the label position, since the primitive assumes

that the label is current.

current_enlarged_label_conflict(Lab1,Lab2,Conflict). - Conflict = 1 if Lab1 and Lab2, at current positions and enlarged by X & Y buffer zone amounts, are in conflict. If not, Conflict = 0.

current_label_overlap(Lab1,Lab2,Overlap). - Overlap = 1 if Lab1 and Lab2, at current positions, are in overlap. If not, Overlap = 0.

determine_area_label_positions(No). - Computes positions for a particular area label (current). These are then retained in FORTRAN. The primitive also returns the number of positions available.

enlarged_label_conflict(Lab1,Pos1,Lab2,Pos2,Conflict). - Conflict = 1 if Lab1 at position Pos1 and Lab2 at position Pos2 are in conflict. If not, Conflict = 0.

find_no_of_area_label_positions(No). - Finds the number of positions available to an area label (current), once these have been computed.

initialise_potential_labels_in_overlap. - Once all label configuration and dimension attributes have been written to the label status file, this primitive determines which labels can potentially overlap each other. See "no_of_potential_labels_in_overlap".

label_overlap(Lab1,Pos1,Lab2,Pos2,Overlap). - Overlap = 1 if Lab1 at position Pos1 and Lab2 at position Pos2 are in overlap. If not, Overlap = 0.

no_of_labels_in_overlap(Label,N). - Finds out how many labels (N) can potentially overlap with Label ("LABEL" file record number).

no_of_potential_labels_in_overlap(Label,N). - Once the initialisation of potential labels in overlap has been performed, the primitive can find out how many labels (N) can potentially overlap with Label ("LABEL" file record number).

3.5 VALIDATION.

valid_feat(Ftype,Fsn). - Confirms that a feature of serial number Fsn, and type Ftype, exists. Ftype can be one of: "0", "point", "1", "line", "2", "area".

valid_font(F). - Confirms that the font number F is not zero.

valid_label(N). - Confirms that the Nth "LABEL" (label status register) record is valid.

valid_name(N). - Confirms that the Nth "NAME" (name index) record is valid.

valid_name_def(N). - Confirms that the Nth "NAME_DEF" (name index) record is valid.

valid_type(Ftype,Class). - Confirms that the feature type Class is 0 for a point, 1 for a line and 2 for an area. Ftype can be one of: "0", "1", "2", "point", "line", "area". Class is always returned as "0", "1" or "2".

3.6 TEXT PROCESSING.

not_available. - **Prints the words "not available yet" on** the screen. This can be used in place of a primitive which has not been implemented.

new_lines(N). - Prints N blank lines on the screen.

dot. - Writes a dot to an output file.

get_ans(X). - Gets a number from the keyboard. Used in the selection of menu options.

put_text_into_mem(N). &
put_text_into_mem. - Puts the ASC-II values of the letters in the name (or current name) in the "NAME" file (record N) into FORTRAN memory (thousand element integer array).

3.7 USEFUL PROLOG PREDICATES.

append(List1,List2,List3). - Appends List1 and List2 together and outputs to List3.

convert(No,Units,Out). - Converts a number (No), from Units (m, mm, inches, cm, km, miles) into Out metres on the ground.

convert(M,m,P,pixels). - Converts M metres, on the ground, into P pixel units in the raster image.

convert(P,pixels,M,m). - Converts P pixel units into M metres on the ground.

extract_element(N,List,Element). - Extracts the Nth Element from a List.

get_list(N,List). - Reads a List of integers, of length N, into PROLOG from FORTRAN memory.

loop(I,Start,End). - If forced to backtrack, will do so (End-Start+1) times. "I" keeps track of the current index number, (an integer number between Start and End). This primitive is similar to a DO loop in FORTRAN.

member(Element,List). - Confirms that Element is a member of List.

minimum(List,Element). - Finds the smallest element in a List.

pull_off(Element,List,N). - Pulls off individual elements of a list on being forced to backtrack N times.

put_list(List). - Writes a List of integers from PROLOG into FORTRAN memory.

read_from_memory(No,Mem). - Reads the contents of array element Mem held in a thousand element FORTRAN array. The contents are returned as No in PROLOG.

reverse(List,Rev_list). - Reverses a list and outputs this to Rev_list.

select(Element,List,Remain_list). - Selects an element from a list and returns the remainder. Can be used in a recursive predicate instead of "member".

select_first(Element,List,Remain_list). - Like "select", except it does not backtrack.

sort(List,Slist). - POPLOG supplied predicate. Sorts a List into alphabetic and/or numeric order (Slist).

write_to_memory(No,Mem). - Writes the PROLOG integer No to an array element Mem held in a thousand element FORTRAN array.

REFERENCES

ANDERSSON, L.E. and WHITAKER, E.A., NASA Catalogue of Lunar Nomenclature, NASA Reference Publication 1097, October 1982.

ANON, O.S., Dicussions at the Ordnance Survey Head Quarters at Southampton.

BASOGLU, U., "A new approach to automated name placement", Proc. Auto-Carto 5, Crystal City, Virginia, USA, 1982, pp103-112.

BASOGLU, U., "An in-depth study of automated name placement systems", Doctoral Dissertation, University of Wisconsin, Madison, 1984.

BOOTHROYD, D., "Computers that learn for themselves", Electronic Times, 11 February 1988, pp18.

BOWELL, E., Ordnance Survey, Private Communication, 1988.

BURROUGH, P.A., Principles of Geographic Information Systems for Land Resource Assessment, Oxford Science Publications Monographs on Soil and Resources Survey No. 12, 1986.

COOK, A.C., "A method of automatically annotating maps", in Research Based on Ordnance Survey Small-scales Digital Data, edited by M.Visvalingam. The University of Hull. Cartographic Information Systems Research Group, 1986 pp52-60.

CROMLEY, R.G., "A spatial allocation analysis of the point annotation problem", Proc. Second Int'l Symposium on Spatial Data Handling, Seattle, Washington, IGU, July 1986, pp38-49.

DAVIS, J.C. and MCCULLAGH, M.J. (Editors), "The Surface II graphics System", Display and analysis of spatial data, NATO Advanced Study Institute, 1975, pp244-266.

DE SIMONE, M., "Automatic structuring and feature recognition for large scale digital mapping", Auto Carto London, Vol. 1, pp86-95, 1986.

DETTMER, R., "The artful transputer", Electronics and Power, August 1986, pp578-582.

DOUGLAS, D.H. and PEUCKER, T.K., "Algorithms for the reduction of the number of points required to represent a digitised line or its caricature", The Canadian Cartographer, 10(2), pp112-122, 1973.

FREEMAN, H. AND AHN, J., "AUTONAP - An expert system for automatic map name placement", Proc. First Int'l. Symposium on Spatial Data Handling, Zurich, Switzerland, IGU, August 1984, Vol. 2, pp544-569.

FREEMAN, H., "The automatic labelling of geographic maps - a problem in computer aesthetics", Graphics Interface 85, pp273-281.

GRAHAM, N., "Introduction to Pascal", 1984, 2nd Edition, pp131.

GREGGAINS, A., "A strategy for name placement", Unpublished, University of Cambridge Computer Laboratory, September 1982.

GREGGAINS, A., "Automated Name placement", Cartographica, 19(2), Monograph 28, 1982, pp133-135.

GUEST, J.E. and GREELEY, R., Geology on the Moon, The Wykeham Science Series, London, 1977.

HADLEY, C., Ordnance Survey, Private Communication, 1986.

HARRINGTON, S., "Computer Graphics A Programming Approach", McGraw Hill, International editions Computer Graphics Series, 1987.

HAYWOOD, P.E., "The Ordnance Survey 1:625000 Database: General Principles and Data Structures". Ordnance Survey document, August 1984.

HAYWOOD, P.E., "Digital 1:50 000 Data Model", Research Based on Ordnance Survey Small-scales Digital Data, edited by M.Visvalingam. The University of Hull. Cartographic Information Systems Research Group, 1986, pp16-23.

HIRSCH, S.A., "An algorithm for automatic name placement around point data", The American Cartographer, 9(1), 1982, pp5-17.

HIRSCH, S.A. and GLICK, B.J., "Design issues for an intelligent names processing system", Proc. Auto-Carto 6, Vol. 1, 1983, pp337-346.

HMSO, "The Department of Transport Manual", 3rd Edition, 1979, Seventh impression 1985.

IMHOF, E., "Positioning names on maps", The American Cartographer, 2(2), 1975, pp128-144.

JONES, C.B., "Cartographic name placement with PROLOG", Unpublished paper, 1987, Polytechnic of Wales.

KEATES, J.S., "Cartographic Design and Production", Longman, London, 1973, pp176-211.

KING, N, Ordnance Survey, Private Communication, 1987,1988.

KOWALSKI, R., "Logic for problem solving", The computer science library. Artificial intelligence series 7, 1979.

LANGRAN, G.E., & POIKER, T.K.,"Integration of name selection and name placement", Proc. 2nd Int'l. Symposium on Spatial Data Handling, Seattle, Washington, IGU, July 1986, pp50-64.

LAWRENCE, G.R.P., "Cartographic methods", Methuen, London, 1971, pp45-70.

MARK, D.M. and GOODCHILD, M.F., "On the ordering of Two-Dimensional Space: Introduction and relation to Tesseral Principles", Spatial data Processing using Tesseral Methods, NERC Unit for thematic Information Systems, 1986.

MAYES, M.H., "O.S. Small Scales Digital Development - Past, Present and Future", Research Based on Ordnance Survey Small-scales Digital Data, edited by M.Visvalingam. The University of Hull. Cartographic Information Systems Research Group, 1986, pp2-16.

MCMASTER, R.B., "Automated line generalization", Cartographica, 24(2), Monograph 37, 1987, pp74-111.

MONMONIER, M., "Computer Assisted Cartography Principles and Practice, Eaglewood Cliffs, NJ, Prentice hall, 1982, pp3.

MOWER, J.E.,"Name placement of point features through constraint propagation", Proc. Second Int'l Symposium on Spatial Data Handling, Seattle, Washington, July 1986, pp65-73.

NAGY, G., and WAGLE, S., "Geographic data processing", Computing Surveys, 11(2), June 1979, pp139-181,

O.S. Rules, Ordnance Survey internal documents.

PAULFRAMAN, D., Technology (G.B.), 1984, 8(12), pp29.

PEUCKER, T.K. & CHRISMAN, N., "Cartographic data structures", The American Cartographer, 2(1), 1975, pp55-69.

PFEFFERKORN, C., BURR, D., HARRISON, D., HECKMAN, B., ORESKY, C., and ROTHERMEL, J., "ACES: A cartographic expert system", Proc. Auto-Carto 7. Washington DC, 1985, pp399-407.

RHIND, D., "Drafting digital maps", The Times Higher Education Supplement, 1st January 1985, pp iii - iv.

RHIND, D., "Digital mapping data standards: an international persepctive", Survey and mapping 85, The 2nd UK National Land Surveying and mapping Conference and Exhibition, University of Reading, 1985.

ROBBINS, "Cartographic publishing with reference to the making and marketing of road maps and atlases", Survey and mapping 85, The 2nd UK National Land Surveying and mapping Conference and Exhibition, University of Reading, 1985.

ROBINSON, J.A., "A machine-orientated logic based on the resolution principle", J. ACM, Vol. 12, pp23-41, 1965.

STERLING, L. and SHAPIRO, E., "The Art of Prolog: Advanced Programming Techniques", MIT Press, Cambridge, Mass., 1986.

VAN ROESSEL, J.W., "An algorithm for locating candidate labelling boxes within a polygon", Auto-Carto 8, Proc. of the Eighth International Symposium on Computer-Assisted Cartography, Baltimore, Maryland, USA, March 29 - April 3, 1987, pp689-699.

WATERMAN, D.A., "A Guide to Expert Systems", Addison-Wesley, Reading, Mass., 1985.

WINSTON, P.H., and HORN, B.K.P., "LISP", Addison-Wesley, 1981.

YOELI, P., "The logic of automated map lettering", The Cartographic Journal, 9(2), December 1972. p99-108.

ZORASTER, S., "Integer programming applied to the map label placement problem", Cartographica, 23(3), 1986, pp16-17.

ZORASTER, S. and Bayer, S., "Practical experience with a map label placement program", Auto-Carto 8, Proc. of the Eighth International Symposium on Computer-Assisted Cartography, Baltimore, Maryland, USA, March 29 - April 3, 1987. pp701-708.